PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	An effective hybrid video deinterlacing algorithm
Auteur: Author:	Hossein Mahvash Mohammadi
Date:	2009
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Mahvash Mohammadi, H. (2009). An effective hybrid video deinterlacing algorithm [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8288/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/8288/
Directeurs de recherche: Advisors:	
Programme: Program:	Non spécifié

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

AN EFFECTIVE HYBRID VIDEO DEINTERLACING ALGORITHM

Hossein Mahvash Mohammadi

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÉSE PRÉSENTÉE EN VUE DE L'OBTENTION

DU DIPLÔME DE PHILOSOPHIAE DOCTOR (PH.D.)

(GÉNIE ÉLECTRIQUE)

April 2009

© Hossein Mahvash Mohammadi, 2009

Library and Archives Canada

Published Heritage Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et Archives Canada

Direction du Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-49421-9
Our file Notre référence
ISBN: 978-0-494-49421-9

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée

AN EFFECTIVE HYBRID VIDEO DEINTERLACING ALGORITHM

Présentée par: Hossein Mahvash Mohammadi

en vue de l'obtention du diplôme de Philosophiae Doctor

a été dûment acceptée par le jury d'examen constitué de :

M. DAVID Jean-Pierre, Ph.D. président

M. SAVARIA Yvon, Ph.D., membre et directeur de recherche

M. LANGLOIS Pierre, Ph.D., membre et co-directeur de recherche

M. BOYER Francois-Raymond, Ph.D., membre

M. NICHOLAS ROZON Côme, Ph.D., membre externe

Dedicated to my wife

Maryam

ACKNOWLEDGEMENTS

I would like to thank my research supervisor, Professor Yvon Savaria, for his invaluable support, supervision and useful suggestions throughout my PhD. I also would like to thank my co-supervisor, Professor Pierre Langlois, for his technical support and guidance. I am grateful to all staff and graduate students who helped me during my research and education. I would like to thanks my family for their help and support; specially I am deeply grateful to my wife for her encouragement and love during all these years. Finally, I gratefully acknowledge the financial support I received from NSERC and Gennum Corporation.

ABSTRACT

HDTV system networks are expanding, but most TV systems still use traditional interlaced scans. In order to display the traditional TV signals on HDTV, deinterlacing and scan up conversion are applied. Even if all traditional TV systems were replaced with HDTV system, deinterlacing would be required to convert the video sequences that were originally recorded in interlaced scan to a progressive scan. Deinterlacing can improve the video quality by increasing the vertical resolution of the picture and by removing interlacing artifacts. Nowadays, the greatest challenge for deinterlacing is balancing the trade-off between implementation complexity and a reasonable image quality.

In this thesis we propose an effective hybrid deinterlacing algorithm in which a motion compensated algorithm is combined with an edge-based method based on the reliability of motion vectors. A five-field motion compensated algorithm that works by using the amount of vertical motion within the reference fields is proposed to achieve the maximum vertical resolution improvement.

A new measure for motion vector reliability assessment called reverse motion estimation (RME) is also proposed to qualify the motion vectors. The method tries reversing the calculated motion vector to confirm its validity. This measure was used in the hybrid algorithm to recognize unreliable motion vectors and switch to an edge-based method to prevent artifacts. The RME measure has a very low computational complexity compared to the other measures while being more efficient.

A Pattern-Based Directional Interpolation (PBDI) deinterlacing algorithm is introduced. The algorithm can be used in the hybrid method in the case that unreliable motion vectors are encountered. It compares three-pixel patterns in the upper and lower lines to detect twenty one edge directions and to provide clear and smooth edges with less probability chance of misleading edges. Pattern comparison is done by a combination of the sum of absolute differences and a gradient-based difference.

RÉSUMÉ

Les réseaux de télévision supportant la télé à haute définition sont en pleine expansion, mais la plupart des systèmes de télévision utilisent toujours le balayage entrelacé. Afin d'être en mesure d'afficher les signaux de télé conventionnels sur les téléviseurs haute définition, le désentrelacement et la conversion de résolution sont utilisés. Même si tous les systèmes de télé conventionnels étaient remplacés par des systèmes haute définition, le désentrelacement serait tout de même nécessaire pour convertir les séquences vidéo ayant été originalement enregistrées dans un format entrelacé. Le désentrelacement peut améliorer la qualité vidéo en augmentant la résolution verticale et en éliminant des artéfacts causés par l'entrelacement. De nos jours, le plus grand défi réside dans le compromis entre la complexité de l'implémentation et la qualité d'image.

Dans la présente thèse, nous proposons un algorithme de désentrelacement hybride efficace où un algorithme à compensation du mouvement est associé à un algorithme basé sur les arêtes de manière à s'adapter à la fiabilité des vecteurs de déplacement. Un algorithme à compensation du mouvement à cinq trames tenant compte du déplacement vertical dans les trames de référence de manière à obtenir l'amélioration optimale de la résolution verticale est présenté.

Une nouvelle mesure de la fiabilité des vecteurs de déplacement nommée évaluation du mouvement inverse (RME) est aussi proposée. Cette méthode tente d'inverser les vecteurs de déplacement de manière à confirmer leur validité. Cette mesure est appliquée

à l'algorithme hybride pour reconnaître les vecteurs peu fiables et, le cas échéant, une méthode basée sur les arêtes est utilisée afin d'éviter l'apparition d'artéfacts. La mesure RME a une complexité algorithmique très faible comparée aux autres mesures existantes tout en étant plus efficace.

Un algorithme d'interpolation directionnelle basé sur les patrons (PBDI) est présenté. L'algorithme peut être utilisé dans la méthode hybride lorsque des vecteurs de déplacement peu fiables sont obtenus. Il compare des patrons de trois pixels dans la ligne supérieure et inférieure de manière à détecter vingt et une directions d'arête et d'obtenir des arêtes nettes et lisses avec une plus grande robustesse de détection vis-à-vis des arêtes trompeuses. La comparaison des patrons est effectuée à l'aide d'une combinaison de la somme des différences absolues et d'une différence de gradients.

CONDENSÉ EN FRANÇAIS

1. Introduction

Aujourd'hui, la plupart des systèmes de télévision font usage du balayage entrelacé. L'entrelacement a été à l'origine utilisé sur les signaux vidéo parce qu'il permet de transmettre avec une bande passante réduite une quantité d'information subjective jugée acceptable. Si le taux d'images d'un signal vidéo est inférieur à un certain seuil, un effet de scintillement est observé. L'augmentation du taux d'images au-dessus de ce seuil requiert une plus grande bande passante que les premiers téléviseurs étaient incapables d'afficher. L'entrelacement apportait une solution au problème de scintillement requérant une bande passante de la moitié de celle qui aurait été nécessaire autrement. Ainsi, le balayage entrelacé permettait une réduction d'un facteur de deux de la bande passante requise. Bien que l'entrelacement ne soit plus aujourd'hui la meilleure solution d'un point de vue technique pour l'économie de bande passante, la compatibilité avec ce choix historique est l'une des contraintes du vaste marché de la télévision.

Dans les systèmes de télévision entrelacée, le groupe des lignes paires est tout d'abord affiché, suivi du groupe des lignes impaires, chacun de ces groupes constituant une trame. À l'opposé, dans le cas des systèmes à balayage progressif, les lignes sont affichées dans l'ordre de manière à former une image complète.

L'entrelacement réduit la qualité de l'image par l'introduction de certains artéfacts et complique plusieurs tâches de traitement vidéo, particulièrement les conversions de format [60]. Les deux artéfacts les plus importants de la vidéo entrelacée sont l'effet de

peigne et le clignotement. Lorsqu'un objet est en mouvement, celui-ci se retrouve à des endroits différents de l'image dans deux trames consécutives. Ceci amène un aspect dentelé des arrêtes nommé effet de peigne. Aussi, si une partie d'un objet a une hauteur inférieure à deux lignes d'image, elle se retrouve à n'être affichée que dans une trame sur deux, ce qui cause le clignotement.

Le passage au balayage progressif implique de sacrifier l'investissement passé des consommateurs et des entreprises dans les systèmes de télévision existants. D'un autre côté, le balayage entrelacé produit des artéfacts inacceptables, particulièrement pour les téléviseurs à grand écran tirant avantage des dernières avancées technologiques. Il est donc préférable que les nouveaux systèmes de télévision supportent les normes existantes et aient ainsi les avantages des deux ensembles de normes.

La norme EDTV (*enhanced definition TV*) définit un format à balayage progressif sans modification apportée au nombre de lignes ou au rapport d'aspect. Le système de télévision à haute définition (TVHD) introduit des formats à définition plus élevée que la télévision conventionnelle et allant jusqu'à 1080 lignes ainsi qu'un passage à un rapport d'aspect de 16:9. Les normes TVHD actuelles comprennent des formats à définition verticale de 1080 lignes à balayage entrelacé et progressif ainsi qu'un format à balayage progressif à 720 lignes. Ces formats sont respectivement nommés 1080i, 1080p et 720p.

Le désentrelacement et la conversion de résolution sont inévitables afin de supporter la gamme existante de formats vidéo à définition standard adoptés dans la TVHD. Des méthodes de désentrelacement de divers types sont utilisées pour générer les lignes manquantes de chaque image. Ces méthodes font usage de techniques d'interpolation

spatiale et temporelle afin d'augmenter la résolution verticale des signaux vidéo entrelacés et d'éliminer les artéfacts dus à l'entrelacement. Alors que le désentrelacement génère les lignes manquantes, la conversion de résolution est utilisée pour convertir le taux d'images, la taille des images et le rapport d'aspect à ceux des formats TVHD.

2. Principes et classification des méthodes de désentrelacement vidéo

2.1. Discussion théorique

Selon le critère de Nyquist, tout signal limité en bande et échantillonné peut être reconstitué parfaitement si le taux d'échantillonnage est de plus du double de la fréquence maximale du signal [25]. L'entrelacement est une procédure de décimation par laquelle le nombre d'échantillons suivant l'axe vertical est réduit de moitié (figure 2.1). La fréquence spatiale maximale d'un signal vidéo à balayage progressif est le double de celle d'un signal vidéo entrelacé.

Selon une autre perspective, l'entrelacement peut aussi être considéré comme une procédure de décimation dans le domaine temporel. La Figure 2.2 montre les données d'une séquence vidéo dans le domaine temporel. Si ces données sont représentées progressivement, chaque instantané de la scène comprend tous les échantillons à chacune des positions de l'écran. L'entrelacement enlève les données de chaque pixel dans une trame sur deux, ce qui correspond bien à une décimation dans le domaine temporel.

Dans les situations où l'entrelacement ne satisfait pas le critère de Nyquist dans le domaine spatial ou temporel, même si le meilleur algorithme de désentrelacement est appliqué, il n'est pas possible de retrouver parfaitement les pixels manquants.

Étant donnée la corrélation spatiale et temporelle interne aux données vidéo, tous les désentrelaceurs font usage de techniques d'interpolation spatiale et temporelle afin de retrouver les lignes manquantes. L'interpolation spatiale n'améliore pas la résolution verticale des images mais peut éliminer certains artéfacts de l'entrelacement. Pour sa part, l'interpolation temporelle peut améliorer la résolution verticale dans les régions statiques de l'image, mais elle produit des artéfacts dans les régions de mouvement.

Relativement aux techniques spatiales et temporelles d'interpolation, les méthodes de désentrelacement sont généralement classées en quatre catégories: intra-trame, inter-trames, adaptatives au mouvement et à compensation du mouvement.

2.2. Méthode de désentrelacement intra-trame

Les méthodes de désentrelacement intra-trame font usage de techniques d'interpolation spatiale afin de retrouver les lignes manquantes. La valeur de chaque pixel est calculée à partir de celle des pixels voisins dans la même trame. Les méthodes intra-trame ne peuvent généralement pas améliorer la résolution verticale des séquences vidéo entrelacées. En particulier, ces méthodes de désentrelacement sont de faible qualité dans le cas des arrêtes horizontales où l'intensité est modifiée de manière importante dans la direction verticale. En raison des faibles exigences en termes de complexité algorithmique et de la mémoire nécessaire, ces méthodes sont les plus économiques [8].

L'interpolation linéaire verticale (BOB) et la répétition des lignes sont les méthodes les plus économiques. Dans le cas de l'interpolation linéaire verticale, chaque pixel manquant est calculé en effectuant la moyenne du pixel supérieur et du pixel inférieur. Ceci cause un flou et il s'ensuit inévitablement une perte de résolution de l'image. Le flou

est un désavantage inhérent aux filtres à moyenne utilisés par la plupart des méthodes intra-trame [47]. Dans le cas de la répétition des lignes où chaque ligne est obtenue en copiant la précédente, l'image résultante a un aspect crénelé et pixélisé. La résolution n'est pas améliorée [22].

Les méthodes de désentrelacement basées sur les arrêtes produisent, parmi les méthodes intra-trame, les résultats ayant la meilleure qualité. Elles tentent de déterminer la direction des arrêtes passant par chaque pixel manquant et l'interpolation est alors effectuée suivant cette direction. La méthode ELA (*Edge-based line averaging*) est la plus connue de cette classe [14]. Les méthodes basées sur les arrêtes fonctionnent bien dans les régions où se trouvent des arrêtes dominantes puisque la direction de celles-ci peut être déterminée de manière fiable. Dans les régions d'arrêtes horizontales ou de haute fréquence spatiale, un résultat de moins bonne qualité peut être obtenu en raison d'une mauvaise estimation de la direction de l'arête. Les arrêtes à direction trompeuse constituent un autre problème susceptible de réduire l'efficacité des méthodes basées sur les arêtes [75].

La méthode de type Inpainting peut être catégorisée dans la classe des méthodes intra-trame parce que cela utilise la donnée de la trame courante [2]. La méthode Inpainting permet de recouvrer la surface cachée par superposition en utilisant une technique d'interpolation des données à partir de l'information locale de la trame courante. L'interpolation est basée sur l'utilisation de la méthode elastica qui se base sur la métrique du niveau d'énergie de la ligne de niveau de la structure de la trame.

2.3. Désentrelacement inter-trames

Les méthodes inter-trames font usage de techniques d'interpolation temporelle afin de calculer les lignes manquantes, tirant ainsi avantage de la corrélation temporelle entre les trames d'une séquence. Elles donnent les résultats ayant la meilleure qualité dans les régions statiques mais nécessitent d'emmagasiner temporairement au moins une trame complète. Elles améliorent la résolution de l'image dans les régions statiques mais, en raison du déplacement des objets dans toute paire de trames consécutives, elles créent l'effet de peigne, l'un des artéfacts les plus agaçants pour l'œil humain. L'appariement des trames (*weave*), l'interpolation linéaire inter-trames et le filtre médian vertical-temporel sont des méthodes de désentrelacement inter-trames populaires.

2.4. Désentrelacement adaptatif au mouvement

Les méthodes adaptatives au mouvement allient les avantages des méthodes intra-trame et inter-trames. Elles font usage de l'interpolation spatiale dans les régions de mouvement et de l'interpolation temporelle dans les régions statiques. Un algorithme de détection du mouvement est nécessaire afin de reconnaître les régions de mouvement et il s'agit du composant le plus important des méthodes adaptatives. Un tampon mémoire est nécessaire pour emmagasiner les trames utilisées pour la détection du mouvement et l'interpolation temporelle. Certaines méthodes adaptatives considèrent aussi une région frontière dans laquelle la détection du mouvement n'est pas fiable. La manière de gérer cette région frontière est une problématique importante de ces méthodes.

Divers algorithmes de détection du mouvement ont été proposés dans la littérature [32][43][37][39]. Ils calculent la différence de luminosité entre certains pixels voisins

dans les trames voisines et comparent cette différence à un seuil dans le but de déterminer la présence ou l'absence de mouvement. Deux problèmes importants des algorithmes de détection du mouvement sont les faux négatifs et les faux positifs. Dans le cas d'un faux négatif, la région est faussement considérée statique et l'interpolation temporelle est donc utilisée à tort, ce qui dégrade la qualité de l'image. À l'inverse, dans le cas d'un faux positif, l'interpolation spatiale est utilisée alors que l'interpolation temporelle devrait l'être. Ceci ne crée pas d'artéfact important mais la résolution verticale de l'image n'est pas améliorée [18].

2.5. Désentrelacement à compensation du mouvement

Les méthodes à compensation du mouvement sont les plus puissantes et les plus complexes. Elles tentent de supprimer virtuellement le mouvement des séquences vidéo de manière à profiter des avantages de l'interpolation temporelle autant dans les régions de mouvement que dans les régions statiques [22]. Un algorithme d'estimation du mouvement est appliqué pour calculer le déplacement de tout pixel, de tout bloc ou de l'image complète. L'estimateur de mouvement conventionnel calcule le vecteur de déplacement entre chaque bloc et le bloc le plus similaire dans les trames voisines.

La qualité des méthodes à compensation du mouvement dépend grandement de la précision de l'estimateur [27]. La méthode de recherche et le critère pour l'appariement sont deux facteurs affectant de manière prédominante cette précision.

Les algorithmes conventionnels d'estimation du mouvement recherchent dans les trames voisines le bloc le plus similaire - ou bloc à appariement optimal - pour chaque bloc de la trame courante (figure 2.5). L'algorithme calcule la différence absolue

moyenne (MAD) ou la somme des différences absolues (SAD) entre le bloc courant et les candidats se trouvant dans les trames voisines à l'intérieur d'une fenêtre de recherche.

Diverses méthodes de recherche sont proposées dans la littérature [3][63][61][31][24][16][44][48][78][49]. La méthode exhaustive donne les meilleurs résultats mais présente la plus grande complexité algorithmique, ce qui rend très difficile son application en temps réel. Dans cette méthode, tous les blocs situés dans la fenêtre de recherche sont comparés au bloc courant.

Il existe des méthodes plus rapides que la méthode exhaustive. Elles réduisent le nombre de blocs de la fenêtre de recherche devant être comparés mais posent l'hypothèse que la MAD augmente de manière monotone avec l'augmentation de la distance par rapport au bloc à appariement optimal. Cette hypothèse n'est pas toujours vraie et la recherche peut converger vers un optimum local. Les algorithmes Cross *search* et *Three-Step search* sont tous deux populaires dans cette classe.

Dans le cas de la méthode à compensation de mouvement la plus simple, l'interpolation linéaire verticale est tout d'abord appliquée à la trame courante en guise de pré-filtrage. Ensuite, l'estimateur de mouvement recherche le bloc à appariement optimal dans la trame précédente. Les lignes manquantes dans la trame courante sont remplacées par les lignes correspondantes du bloc à appariement optimal dans la trame précédente.

Les méthodes à compensation de mouvement peuvent générer des artéfacts si les vecteurs de déplacement sont erronés. Il existe des situations particulières où l'estimation de mouvement conventionnel ne peut fournir des vecteurs de déplacement fiables. Par exemple, les estimateurs de mouvement conventionnels peuvent ne pas fonctionner

lorsque la séquence vidéo présente une rotation ou une mise à l'échelle, lorsqu'un objet disparaît puis reparaît, lorsqu'un objet sort de la fenêtre de recherche, ou en cas de déplacement de distance inférieure à un pixel. Les méthodes de recherche rapides peuvent aussi donner des vecteurs de déplacement erronés.

Une nouvelle classe de méthodes de désentrelacement hybrides allie des éléments de plusieurs classes. Dans plusieurs travaux, les méthodes à compensation de mouvement et intra-trame ont été combinées. L'utilisation de l'une ou de l'autre dépend de la fiabilité des vecteurs de déplacement calculés. En effet, ces méthodes hybrides font usage de la compensation du mouvement lorsque les vecteurs de déplacement sont fiables et de l'interpolation spatiale autrement. Ceci nécessite l'utilisation d'une métrique qualifiant la fiabilité de ces vecteurs. Les méthodes hybrides permettent, comme la compensation du mouvement, d'améliorer la résolution verticale, mais contrairement à celle-ci, ne créent pas d'artéfacts dus aux vecteurs de déplacement erronés. Le tableau 2.1 résume les problématiques pertinentes aux diverses méthodes de désentrelacement et en fournit une comparaison qualitative.

3. Problème de l'évaluation de la qualité de l'image

Des métriques objectives et subjectives peuvent être utilisées pour évaluer la performance d'un désentrelaceur. La caractérisation subjective nécessite souvent trop de temps pour être pratique et des facteurs humains tels que l'âge ou la culture peuvent influencer la qualité perçue.

L'erreur quadratique moyenne est largement utilisée pour la caractérisation du traitement d'image. Cependant, cette métrique est souvent critiquée pour son manque de

corrélation avec le système visuel humain. Ce système visuel a une sensibilité variant avec la fréquence, ce qui n'est pas considéré par l'erreur quadratique moyenne, ni d'ailleurs par le PSNR (peak signal to noise ratio). Autre point non considéré, le système visuel humain est plus sensible au flou qu'à d'autres types de bruit. Il semble que le système visuel utilise une comparaison de similarité structurelle des images, mais la plupart des métriques actuelles effectuent plutôt une comparaison différentielle. La définition d'une métrique objective de la qualité d'une image reste une question ouverte.

4. Contributions

Nous proposons une méthode de désentrelacement hybride à compensation du mouvement. La méthode proposée fait usage de deux algorithmes également proposés: un algorithme à compensation du mouvement à cinq trames (FFMC) et un algorithme intra-trame nommé interpolation directionnelle basée sur les patrons (PBDI). Une nouvelle méthode de mesure de la qualité des vecteurs de déplacement nommée estimation de mouvement inverse (RME) est aussi proposée. La méthode hybride utilise la RME pour déterminer la fiabilité des vecteurs de déplacement, après quoi, les pixels sont calculés à l'aide de FFMC si les vecteurs sont fiables ou de PBDI autrement. Le détail des algorithmes proposés est donné dans ce qui suit.

4.1. Méthode à compensation du mouvement à cinq trames (FFMC)

Dans la méthode à cinq trames proposée, les deux trames précédentes et les deux suivantes sont utilisées comme trames de référence pour l'estimation du mouvement. Un

pré-filtrage par interpolation linéaire des lignes est appliqué dans la trame courante et les suivantes. La méthode de recherche exhaustive est utilisée afin de trouver un bloc à appariement optimal dans chaque trame de référence. Parmi ces blocs, celui pour lequel la somme des différences absolues (SAD) est minimale est sélectionné pour le calcul des vecteurs de déplacement. L'erreur d'appariement est définie comme la SAD entre un bloc et son bloc à appariement optimal comme montré par l'équation (3.2).

Si le déplacement vertical d'un objet entre deux trames consécutives est d'un nombre impair de lignes, les lignes existantes de l'objet dans la trame courante restent inchangées dans toutes les trames de référence. Cette situation est montrée à la figure 3.1.a, dans laquelle, les lignes existantes de la lettre majuscule A restent les mêmes dans toutes les trames de référence. Ainsi, le bloc à appariement optimal peut être trouvé dans chacune des trames de référence puisque les pixels existants de ce bloc sont les mêmes. Dans cette situation, la résolution verticale ne peut pas être améliorée étant donné que des lignes calculées lors du pré-filtrage ou d'un désentrelacement préalable sont utilisées pour les lignes manquantes. Seules les lignes calculées obtenues en recopiant des lignes existantes permettent d'améliorer la résolution verticale.

Si le déplacement vertical d'un objet entre deux trames consécutives est d'un nombre pair de lignes, les lignes existantes du bloc dans la trame courante correspondent aux lignes manquantes de la trame précédente ou suivante. Elles correspondent aussi aux lignes existantes des trames précédente et suivante de même parité. Dans ce cas, le bloc à appariement optimal devrait être trouvé dans la trame précédente ou suivante de même parité. Ceci est montré à la figure 3.1.b.

En présence d'un déplacement vertical inférieur à un pixel tel que montré à la figure 3.1.c, si ce déplacement est de seulement un demi pixel par trame, le déplacement total entre la trame courante et la précédente ou la suivante de même parité sera d'un pixel, mais les lignes existantes de la trame courante seront manquantes. Un plus grand nombre de trames de référence serait nécessaire pour que les lignes existantes réapparaissent et que le bloc à appariement optimal puisse être retrouvé de manière précise. Ceci amène une augmentation de la complexité algorithmique. Cependant, si le bloc à appariement optimal est trouvé dans la trame précédente ou suivante de même parité, les lignes existantes de ce bloc correspondent aux lignes manquantes de la trame courante et suivantes, et peuvent donc être utilisées pour améliorer la résolution verticale.

Lorsque le bloc à appariement optimal est trouvé, l'estimateur de mouvement retourne l'erreur calculée selon l'équation 3.2, les vecteurs de déplacement et l'indice de la trame de référence où il a été trouvé.

La figure 3.2 montre un ordinogramme de la méthode proposée. Si le bloc à appariement optimal est trouvé dans une trame de même parité avec un nombre impair de lignes de déplacement par trame ou dans une trame de parité inverse, les lignes manquantes dans la trame courante sont remplacées par les lignes correspondantes de ce bloc. Dans ce cas, les vecteurs de déplacement sont plus fiables puisque le bloc à appariement optimal et son bloc correspondant ont les mêmes lignes existantes et manquantes, mais la compensation de mouvement calcule les lignes manquantes à partir de lignes obtenues par désentrelacement préalable ou par pré-filtrage, puisqu'elles ne sont pas présentes dans la trame de référence.

Si le bloc à appariement optimal est trouvé dans la trame précédente ou suivante de même parité et que le déplacement vertical par trame est d'un nombre de lignes pair, alors le déplacement vertical total est divisible par quatre. Dans ce cas, les lignes manquantes du bloc dans la trame courante sont remplacées par les lignes existantes du bloc dans la trame précédente ou suivante dont la position est obtenue en divisant le vecteur de déplacement par deux. Il s'agit de la meilleure situation, puisqu'il est alors possible d'améliorer la résolution verticale. Comme montré à la figure 3.2.b, dans ce cas, les lignes existantes pour un bloc dans la trame courante et son bloc à appariement optimal sont les mêmes. Ceci permet une détermination plus fiable du bloc à appariement optimal. De plus, les lignes manquantes sont les lignes existantes dans les trames de parité opposée avec la moitié du déplacement vertical, ce qui rend possible l'amélioration de la résolution verticale.

L'utilisation des trames de même parité pour l'estimation du mouvement peut causer des artéfacts en cas de mouvement rapide ou non uniforme. La figure 3.3 montre trois images consécutives de la séquence « Table Tennis ». L'arrière-plan de ces images est statique et la balle de tennis se déplace rapidement. Les images n et $n+2$ ont le même arrière plan pour les blocs mis en évidence et le bloc à appariement optimal du bloc à l'image n se trouve à la même position à l'image $n+2$.

Maintenant, si le bloc à déplacement de moitié dans l'image $n+1$ est utilisé pour la compensation du mouvement, un artéfact est introduit. Ceci est dû au fait que la balle n'apparaît pas dans le bloc qui nous intéresse dans les images n et $n+2$ mais y apparaît à l'image $n+1$. La figure 3.3.d montre l'artéfact résultant pour la trame n.

Le même type d'artéfact peut se produire en cas de mouvement non uniforme. La figure 3.4 montre cette situation où le mouvement de la balle n'est pas uniforme. En effet, elle descend durant les deux premières images mais remonte à la suivante. Comme montré aux figure 3.4.a et 3.4.c, le bloc à appariement optimal pour celui contenant la balle à l'image *n* se trouve dans la prochaine image de même parité, soit *n+2*. Maintenant, si le bloc à déplacement de moitié dans l'image *n+1* est utilisé pour la compensation du mouvement, l'artéfact montré à la figure 3.4.d est observé dans l'image désentrelacée *n* puisque le bloc contient des pixels d'arrière-plan.

Afin d'éliminer ce type d'artéfact, la méthode proposée comprend une vérification supplémentaire avant que le bloc à déplacement de moitié ne soit utilisé pour la compensation du mouvement. Elle implique le calcul de la somme des différences absolues entre le bloc de la trame courante et le bloc à déplacement de moitié dans la trame de parité inverse. Ce bloc n'est utilisé que si la valeur ainsi calculée est inférieure à un seuil. Dans le cas contraire, les lignes manquantes sont remplacées par les lignes du bloc à appariement optimal obtenues par désentrelacement préalable ou pré-filtrage.

Le seuil est choisi en fonction de l'erreur du bloc à appariement optimal. Cette erreur indique à quel point ce bloc et son bloc correspondant dans la trame courante sont corrélés. Cette corrélation doit être conservée lorsque le bloc à déplacement de moitié est utilisé pour la compensation du mouvement. Cependant, étant donné que ce dernier se trouve dans la trame de parité opposée, les lignes existantes et manquantes ne sont pas similaires à celles du bloc dans la trame courante. Pour cette raison, l'erreur entre le bloc de la trame courante et le bloc à déplacement de moitié peut être supérieure à l'erreur du

bloc à appariement optimal. Pour cette raison, le seuil correspond à cette erreur multipliée par un coefficient, comme montré à l'équation 3.4.

4.2. Méthode directionnelle basée sur les patrons (PBDI)

La plupart des méthodes de désentrelacement basées sur les arrêtes comparent des pixels individuels dans la ligne supérieure et inférieure afin d'estimer la direction de l'arête. Dans les régions à haute fréquence spatiale et présentant des patrons complexes, ceci peut fausser la détection de la direction de l'arête. Afin d'améliorer l'algorithme réalisant cette détection, nous proposons la technique PBDI où la comparaison est effectuée entre des patrons de pixels au lieu de pixels individuels. Un patron à la position (i, j) est défini comme un vecteur de trois pixels horizontalement adjacents dans l'équation 4.3.

Deux fenêtres de recherche sont définies, soit l'une dans la ligne immédiatement supérieure et l'autre dans la ligne immédiatement inférieure par rapport au pixel manquant. Ces fenêtres de recherche ont une largeur de neuf pixels et sont centrées horizontalement par rapport au pixel manquant. La figure 4.5 illustre cette situation. Le pixel manquant est à la position (i, j). Deux patrons sont montrés, soit un dans chaque fenêtre de recherche, et tous deux définissent l'angle d'une arête.

Sept patrons sont possibles dans chaque fenêtre de recherche. $P(i\text{-}3, j\text{-}1)$ à $P(i\text{+}3, j\text{-}1)$ sont ceux de la fenêtre de recherche supérieure et $P(i\text{-}3, j\text{+}1)$ à $P(i\text{+}3, j\text{+}1)$ ceux de la fenêtre inférieure. Des paires de patrons sont comparées en prenant un patron dans chaque fenêtre. Les deux patrons les plus similaires définissent la direction d'une arête. Dans la figure 4.6, si on suppose que les patrons mis en évidence sont les plus similaires,

la direction de l'arête correspondante serait suivant le segment *a* qui traverse le pixel central de chaque patron.

L'algorithme PBDI augmente grandement le nombre de combinaisons possibles de directions et de positions d'arêtes aux 21 montrées à la figure 4.7. Chaque arête considérée est supposée passer par le pixel central des patrons supérieur et inférieur. De plus, PBDI peut détecter les arrêtes ne traversant pas la ligne j exactement à la position du pixel manquant. La figure 4.8 montre un exemple où l'arête traverse la ligne j à la position $i - 0.5$. Les méthodes conventionnelles basées sur les arêtes qui comparent des pixels individuels n'atteignent pas ce niveau de résolution.

Une inspection de la figure 4.7 montre que toutes les paires de patrons possibles ne sont pas considérées. Pour des patrons de trois pixels et des fenêtres de recherche de neuf pixels, 49 paires sont effectivement possibles. Vingt parmi celles-ci définissent des arêtes qui ne traversent pas le pixel manquant et ne fournissent donc pas d'information pertinente en vue de l'interpolation. Huit autres paires sont aussi exclues afin de réduire le nombre de situations d'arêtes trompeuses, comme ce sera expliqué plus loin. Ceci laisse les 21 paires de patrons utiles montrées à la figure 4.7.

L'algorithme PBDI proposé est robuste par rapport aux arêtes trompeuses, situation se produisant lorsque l'arrière-plan est plus corrélé que l'arrête. Cette robustesse est obtenue en incluant le pixel immédiatement supérieur ou celui immédiatement inférieur par rapport au pixel manquant, ou encore les deux, dans toutes les comparaisons. Afin de satisfaire cette contrainte, seuls trois patrons de chaque fenêtre de recherche sont comparés à tous les patrons de l'autre fenêtre. Ceci résout le problème des arêtes

trompeuses pour toutes les arêtes traversant les pixels immédiatement supérieur ou inférieur par rapport au pixel manquant.

La figure 4.9 montre une arête traversant le pixel supérieur par rapport au pixel manquant. La présence du pixel (i, j-1) dans tous les patrons comparés dans la fenêtre de recherche supérieure a un impact significatif sur le calcul de la similarité. En effet, ceci empêche que les pixels de l'arrière-plan soient considérés comme ayant une plus grande similarité que les pixels de l'arête réelle.

Dans le domaine du codage vidéo, plusieurs critères tels que la somme de différences absolues (SAD) et la corrélation de phase on été proposés afin de trouver les blocs les plus similaires [40]. Dans le cas de l'algorithme PBDI, une combinaison de la SAD et de la différence des gradients (GBD) est utilisée pour la comparaison de patrons. La GBD consiste à calculer la différence d'intensité entre le pixel central et ses deux voisins latéraux. Ces différences sont alors comparées pour chaque paires de patrons considérés. Dans certaines situations, la GBD donne de meilleurs résultats que la SAD. Ceci comprend la comparaison de patrons ayant la même texture mais des niveaux d'illumination différents.

Nous proposons la sélection des patrons les plus similaires par une combinaison linéaire de la SAD et de la GBD. L'erreur entre deux patrons est calculée par l'équation 4.4 où GBD est la différence des gradients et α est un coefficient déterminant l'importance relative de la SAD et de la GBD. P_u et P_l sont respectivement les patrons dans la fenêtre de recherche supérieure et inférieure. Le coefficient α est déterminé de manière empirique. Selon nos essais, les valeurs de α situées dans la plage 0,6 à 0,8

donnent les meilleurs résultats en termes de l'erreur quadratique moyenne pour la plupart des séquences vidéo vérifiées. La SAD pour les patrons $P(i, j\text{-}1)$ et $P(n, j\text{+}1)$, notés P_u et P_l respectivement, est calculée à l'aide de l'équation 4.5.

Pour une paire de patrons à comparer, nous notons dif_l la différence de gauche, soit la différence de luminosité entre le pixel central de chaque patron et son voisin de gauche. De même, nous notons dif_r son équivalent à droite. Les équations 4.6 et 4.7 montrent le détail du calcul de dif_l et dif_r pour les patrons P_u et P_l aux positions respectives $(i, j\text{-}1)$ et $(i, j\text{+}1)$. La GBD pour ces patrons est calculée selon l'équation 4.8.

L'arête définie par les patrons les plus similaires est spécifiée par le décalage horizontal de ces derniers par rapport au pixel manquant. Pour les situations considérées jusqu'ici, les valeurs m et n calculées selon l'équation 4.10 représentent respectivement ce décalage pour les patrons sur la ligne supérieure et inférieure. La condition $|m+n| \leq 2$ garantit que les arêtes détectées traversent la ligne j à une distance d'au plus un pixel par rapport au pixel manquant. Les conditions $|m| \leq 1$ et $|n| \leq 1$ sont utilisées pour exclure les comparaisons pouvant causer des situations d'arêtes trompeuses.

Pour des valeurs acceptables de m et n, la valeur d'interpolation pour le pixel manquant est calculée suivant l'équation 4.11. Cette version de l'équation est spécifique au cas de patrons à trois pixels.

L'intensité finale pour le pixel manquant est la médiane entre $f(i, j\text{-}1)$, $f(i, j\text{+}1)$ et la valeur interpolée. En utilisant un argument géométrique, on peut montrer que les équations suivantes, qui réalisent une simple moyenne des pixels, donnent le même résultat qu'une interpolation bilinéaire. Le pixel interpolé est soit aligné avec les pixels

sélectionnés pour l'interpolation directionnelle ou bien il se trouve exactement entre les segments de droite reliant les coins du parallélogramme d'interpolation.

4.3. Méthode hybride à évaluation inverse du mouvement

La meilleure amélioration de la résolution verticale est obtenue par le désentrelacement à compensation du mouvement lorsque l'estimation du mouvement est fiable. Cette estimation peut, dans certaines situations, donner des vecteurs de déplacement peu fiables et ainsi être source d'artéfacts. Ceci comprend les situations de déplacement d'une distance inférieure à un pixel, de rotation ou de mise à l'échelle, de déplacement d'un objet hors de la fenêtre de recherche et de masquage d'un objet par un autre. L'alternance entre les lignes existantes paires et impaires complique aussi l'estimation.

Ainsi, les algorithmes d'estimation du mouvement les plus puissants peuvent ne pas être en mesure de fournir des vecteurs de déplacement fiables. Un estimateur de mouvement fournissant des vecteurs de déplacement fiables dans toutes les situations n'est pas possible en pratique étant donné la grande complexité algorithmique impliquée.

Nous proposons ici l'algorithme hybride effectuant le choix entre les méthodes proposées FFMC et PBDI. Cet algorithme base sa décision sur la fiabilité des vecteurs de déplacement de manière à atteindre la résolution verticale maximale tout en évitant la production d'artéfacts. L'évaluation inverse du mouvement (RME) est proposée pour caractériser cette fiabilité. Dans le cas de vecteurs fiables, l'algorithme proposé choisi la compensation du mouvement alors qu'une méthode intra-trame est utilisée dans le cas contraire. La figure 5.1 montre le diagramme bloc de la méthode hybride.

La méthode RME utilise une méthode très simple pour reconnaître les vecteurs de déplacement peu fiables. Lorsque l'estimateur de mouvement trouve un bloc à appariement optimal, l'algorithme RME recherche à l'intérieur de la trame courante le bloc lui étant le plus similaire. Si le bloc original ou l'un de ses proches voisins est obtenu, les vecteurs de déplacement sont considérés fiables.

La figure 5.1 montre les trames n et n+1 pour décrire la méthode RME. On suppose que l'estimateur de mouvement a trouvé le bloc B de la trame n+1 comme bloc à appariement optimal du bloc A. En utilisant la RME, le bloc C est choisi comme étant le plus similaire au bloc B dans la trame courante. Si le bloc B correspond bien au bloc A, la RME doit référer au bloc A ou à l'un de ses proches voisins. La distance entre le bloc original A et le bloc C est utilisée pour caractériser le vecteur de déplacement. Si cette distance est supérieure à un seuil, le vecteur est considéré peu fiable.

La figure 5.2 montre le diagramme bloc de la méthode hybride faisant usage de la RME. Comme première étape, le pré-filtrage par interpolation linéaire est appliqué dans la trame actuelle et les deux suivantes afin de calculer provisoirement les lignes manquantes en vue de l'estimation du mouvement. Ensuite, l'estimateur de mouvement recherche le bloc à appariement optimal dans les deux trames précédentes et les deux suivantes. d_1 est le vecteur de déplacement associé au bloc à appariement optimal OMB et m indique la trame où ce bloc a été trouvé. En utilisant la RME, la méthode recherche dans la trame n le bloc le plus similaire à OMB, noté $\mathrm{OMB_{RME}}$. d_2 correspond au vecteur de déplacement lié à $\mathrm{OMB_{RME}}$ et $|| d_1 - d_2||$ est la distance entre le bloc original et celui

donné par la RME. Si la distance est supérieure au seuil *DisThr*, le vecteur de déplacement d_1 est considéré peu fiable.

Les résultats expérimentaux montrent que la RME proposée fonctionne mieux que les autres mesures en termes de PSNR global dans les différentes séquences vidéo. La distribution de fréquences cumulées relative à la RME donnée à la figure 5.6 montre que la méthode RME a moins de la moitié du nombre de blocs ayant une grande erreur (plus de 2500) de la méthode *a posteriori* et moins du quart de la méthode LVS. La RME a aussi une complexité algorithmique inférieure à celle des autres mesures. L'efficacité de la RME est justifiée sur la base d'évaluations objectives et subjectives.

5. Conclusion

Les réseaux de systèmes de télévision haute définition sont en pleine expansion, mais la plupart des systèmes de télévision font toujours usage du balayage entrelacé. Il semble que les systèmes de télévision traditionnels resteront sur le marché pour plusieurs années encore. Il est trop tôt pour éliminer complètement ces derniers en faveur des systèmes à haute définition. Donc, l'utilisation du désentrelacement est inévitable. De plus, même si tous les systèmes de télévision passaient à la haute définition, il serait encore nécessaire de convertir les séquences vidéo originalement enregistrées sous un format à balayage entrelacé. De nos jours, la problématique la plus importante pour le désentrelacement est le compromis entre la complexité de l'implémentation et la qualité du résultat.

Nous proposons une structure hybride où un algorithme à compensation de mouvement à cinq trames est utilisé en vue d'obtenir une amélioration maximale de la résolution verticale. Une simple méthode d'évaluation du mouvement inverse est

proposée afin de caractériser les vecteurs de déplacement. La méthode d'évaluation du mouvement inverse a une complexité algorithmique inférieure aux autres méthodes connues tout en étant plus efficace que celles-ci. L'algorithme PBDI est aussi proposé pour retrouver des arrêtes lisses et définies tout en réduisant les chances de détections de fausses arrêtes. L'algorithme PBDI proposé peut être utilisé dans la structure hybride en cas de détection de vecteurs de déplacement peu fiables.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACS	Adaptive Cross Search
ADCS	Adaptive Dual-Cross Search
AMPDF	Adaptive Minimum Pixel Difference Filter
ASIP	Application Specific Instruction Set Processor
BBGDS	Block-Based Gradient Descent Search
CDS	Conjugate Directional Search
DCS	Dual-Cross Search
dB	Decibel
DFD	Displaced Frame Difference
DIF	Directional Correlation Dependent Interpolation Filter
DOI	Directional-Oriented Interpolation
DS	Diamond Search
EDDI	Edge-Dependent Deinterlacing
EDTV	Enhanced Definition TV
ELA	Edge-Based Line Averaging
EMF	Edge-Based Median Filter
FFMC	Five Field Motion Compensated
FWF	Fuzzy Weighted Filtering
GBD	Gradient-Based Difference
HDTV	High Definition TV

HMC	Hybrid Motion Compensated
IPDC	Interlaced Pixel Distortion Classification
LA	Line Averaging
LVS	Local Vector Smoothness
MAD	Mean Absolute Difference
MPEG	Moving Picture Experts Group
MSE	Mean Square Error
MV	Motion Vector
NID	Novel Intra Deinterlacing
NTSC	National Television System Committee
NTSS	New Three Step Search
OMB	Optimal Matching Block
OSA	Orthogonal Search Algorithm
OTS	One at a Time Search
PAL	Phase Alternating Line
PBDI	Pattern-Based Directional Interpolation
PSNR	Peak Signal to Noise Ratio
RGB	Red Green Blue
RME	Reverse Motion Estimation
SAD	Sum of Absolute Differences
SDV	Spatial Direction Vector
SECAM	Sequentiel Couleur A Memoire

STELA	Spatio-Temporal ELA
TDL	2-D Logarithmic Search
TSS	Three Step Search

LIST OF SYMBOLS

a	Edge direction
b	Misleading edge direction
C	Constant
D	Set of all possible estimates for the true motion vector d
$DisThr$	Distance threshold
d	Motion vector
d_x	Horizontal motion vector
d_y	Vertical motion vector
dB	Decibel
dif_l	Difference between middle pixel and its left
dif_r	Difference between middle pixel and its right
Err	OMB error
f	Pixel intensity
f_k	Current frame
f_m	Reference frame
F_i	Input field
F_o	Output frame
f	Original images
$\hat{f}$	Deinterlaced images
i	Pixel's column

j	Pixel's row
k	Direction with the highest directional correlation
l	Lower pattern horizontal displacement
m	Field number in which OMB was found
MV	Motion vector
n	Frame number
P_u	A pattern in the upper search window
P_l	A pattern in the lower search window
P	Pattern
u	Upper pattern horizontal displacement
$U(.)$	Energy function
x	Spatial coordinates of a pixel
X	Image width
x_b	Block's horizontal coordinate
Y	Image height
Y	Luminance component
y_b	Block's vertical coordinate
α	Coefficient for fast and non-uniform motions
α	Coefficient that determines the relative proportions of SAD and GBD
Λ	Set of spatial positions belonging to a block
σ^2	Variance
σ_d^2	Variance of the difference d - d_i

Chapter 1

Introduction

1.1. Motivation

Interlacing was originally incorporated into video signals to reduce the transmission bandwidth by a factor of two. In an interlaced system, a snapshot of the scene is captured, transmitted and displayed alternating between odd and even fields that contain only odd and even lines, respectively. A snapshot of the scene in progressive scan is called a frame and contains all the lines. If the frame rate of a video stream is less than the specific rate, it causes flickering. Increasing the frame rate needs more bandwidth and early televisions were unable to refresh the display at that rate. Interlacing was used because it reduces the flickering artifact and also it kept the required bandwidth at half of what is needed without interlacing. Although interlacing is not currently the best technical solution to save transmission bandwidth, due to compatibility issues most TV systems still use interlaced scan.

Interlacing reduces the image quality through the introduction of artifacts in the image and it complicates many image-processing tasks, particularly scanning-format conversions [60]. Combing and line twittering are the two main artifacts caused by interlacing. When an object is moving, each snapshot of the object in two sequential fields shows the object in different places. As a result, two sequential fields do not

complement each other perfectly and the object edges are jagged. This effect is known as the combing artifact. If the height of a part of an object is less than two scan lines, it is displayed only in one of two sequential fields and the line twittering artifact occurs.

Interlacing causes other types of artifacts especially in large screen TV systems. The conventional TV systems use from 480 (NTSC) to 576 (PAL and SECAM) scan lines. Scan lines become visible in the large size screen when interlacing is used. This decreases the quality of the video and also causes crawling and flickering artifacts.

The poor quality of traditional TV systems, especially on large size screens, demands the higher resolutions that are supported in the EDTV (enhanced definition TV) and HDTV (High definition TV) systems. EDTV came up with a new standard for digital television in progressive scan without any changes in the number of lines or aspect ratio. HDTV systems support both interlaced and progressive scans, use the MPEG-2 standard for compression and change the aspect ratio to 16:9. The current HDTV standards support a vertical resolution of 1080 lines in interlaced and progressive scan, and 720 lines in progressive scan that are referred to as 1080i, 1080p, and 720p, respectively.

Deinterlacing and scan up conversion are needed to convert the existing variety of interlaced standard formats into HDTV formats. Deinterlacing algorithms use temporal and/or spatial interpolation techniques to generate missing lines in order to convert the interlaced video into progressive form. Scan up conversion is used to change the frame rate and size, and aspect ratio to HDTV formats. Even if all TV systems switched to HDTV systems, deinterlacing would still be needed to convert video sequences originally recorded in interlaced scan to a progressive one. Deinterlacing can improve the video

quality by increasing the vertical resolution of the picture and by removing interlacing artifacts. Nowadays, all HDTVs are equipped with a deinterlacer in order to display traditional TV signals with improved quality.

1.2. Research Problems

In this section we summarize the challenging issues in different kinds of deinterlacing algorithms. Interlacing is a down sampling procedure in which the vertical spatial resolution of image is halved. Deinterlacing method use some spatial and/or temporal interpolation techniques to calculate the missing lines in order to improve the spatial vertical resolution of the image and remove interlacing artifacts. Combing and flickering are two main deinterlacing artifacts. A perfect deinterlacing algorithm should remove these artifacts and improve the vertical resolution. However, in situations where the Nyquist rate is not satisfied, deinterlacing cannot reconstruct the missing lines perfectly.

Intra field methods use spatial interpolation to calculate missing pixels. They are the most cost efficient methods but they cannot improve the vertical resolution in the areas with high spatial frequencies. These methods produce flickering artifacts in these areas. Such artifacts are very annoying to human observers. Edge-based deinterlacing methods produce good quality within the intra field methods. They attempt to recognize the direction of the edge at a missing pixel position and then interpolate it toward that direction.

How to calculate the edge direction and misleading edges are the most important issues in the edge-based methods. Using few neighboring pixels to detect the edge direction may cause misleading or inaccurate edge direction estimation. On the other hand, using additional pixels implies a reliance on data that is further away from the missing pixel, and consequently that can be much less correlated. In areas with low spatial frequency and with homogeneous textures, using a large number of neighboring pixels can improve the edge direction detection. In areas with high spatial frequency, using pixels that are widely separated can lead to incorrect edge direction estimation. The design therefore faces a tradeoff between edge angular resolution and information correlation.

Inter filed deinterlacing methods use temporal interpolation to calculate a missing pixel. This improves the vertical resolution in static areas but produce combing artifact which is very distracting in motion areas. An inter-field deinterlacing algorithm alone cannot usually remove the combing effect, unless it considers the amount of motion within the neighboring fields as is done in motion compensated methods.

Motion adaptive methods combine the advantages of intra field and inter field methods. They usually use spatial interpolation in motion areas and temporal interpolation in static areas. These methods need a motion detection algorithm to distinguish motion and static areas that is the most important component of these methods. Using spatial interpolation in motion areas causes flickering artifacts when the areas contain high spatial frequencies.

Interlacing complicates the motion detection algorithm. Same parity fields or sequential fields are used in motion detectors. Using successive fields need to interpolate the missing pixels that can cause incorrect decisions. Using same parity fields requires more memory buffers and increases the temporal distance between missing pixels used for motion detection. "Missing motion", "false motion" and fuzzy areas are major problems of motion detection algorithms.

Motion compensation methods calculate the displacement of the objects within the neighboring fields in order to use the advantages of temporal interpolation in both static and motion areas. They are the most powerful and most complex deinterlacing methods. The quality of these methods highly depends on the accuracy of motion estimation. They can achieve the maximum possible vertical resolution improvement in case of accurate motion vectors.

Interlacing complicates motion estimation and compensation because of alternation of existing lines between odd and even fields. Most algorithms use pre-filtering to calculate missing lines prior to motion estimation. Any interpolation error caused by pre-filtering may be propagated into the next output frames. This is a common drawback of the motion compensated methods using pre-filtering. Some methods use same parity fields for motion estimation that does not need pre-filtering. These methods may produce some artifacts in the presence of fast or non-uniform motions.

Motion estimation in video sequences that contain zooming, rotation and object deformation either involves too much computational complexity, is not practical or does not exist.

How to calculate missing pixels even with perfect motion estimation is a challenging issue. An accurate motion vector may refer to a block that does not have useful data for deinterlacing. It can happen when an original block and the OMB have the same missing and existing pixels. So how to find the missing pixels of a block presented as existing lines in neighboring fields is a challenging issue especially in presence of non-uniform motions.

Due to unavoidable incorrect motion vectors, hybrid motion compensated algorithms have been proposed. These methods usually switch to an intra field method in case of unreliable motion vector to avoid the artifacts of incorrect motion vectors. The most challenging issue in these methods is how to measure the reliability of the calculated motion vectors. A measure to qualify the motion vector is always desired in most applications using motion compensation.

Finally, the most challenging issue in deinterlacing algorithm is how to balance the trade-off between the computational complexity and a reasonable quality.

1.3. Research objectives

The main objective of this thesis is to introduce an effective deinterlacing algorithm with a reasonable quality and the least possible computational complexity. Based on the research problems and requirements, the detailed objectives for an effective deinterlacing method are as follows:

- To develop a new motion compensated deinterlacing algorithm that will eliminate flickering and combing artifacts introduced by intra and inter

field deinterlacing methods.

- To propose a motion estimation algorithm adapted to interlaced video data, as common motion estimation and compensation techniques are not efficient with interlaced video data.
- To develop a motion compensated technique adapted to interlaced video data to calculate the missing lines in order to achieve the best possible vertical resolution improvement.
- To propose a hybrid method including an intra field method that will remove the artifacts caused by unreliable motion vectors when perfect motion estimation is not feasible.
- To develop an edge-based method in order to provide smooth and sharp edges with less probability of misleading edge. This method will be used when motion vectors are not reliable.
- To propose an effective criterion to evaluate the reliability of the calculated motion vectors in the proposed hybrid method.

1.4. Research Contributions

A hybrid motion compensated (HMC) deinterlacing method is proposed in this thesis [52]. The hybrid method consists of a novel motion compensated method called FFMC [50], an edge-based deinterlacing technique called PBDI [51] and a criterion to evaluate the motion vector reliability called RME [52]. When the motion vectors are calculated by FFMC, RME is used to evaluate the reliability of the calculated motion

vectors. If they are reliable, FFMC uses a proposed technique to calculate missing pixels in order to achieve the maximum vertical resolution improvement. In case of the unreliable motion vectors, PBDI is used to prevent artifacts of wrong motion vectors.

[50] Mahvash Mohammadi H., Langlois J.M. P., and Savaria Y., "A Five-Field Motion Compensated Deinterlacing Method Based on Vertical Motion," *IEEE Transactions on Consumer Electronics*, Vol. 53, No. 3, Aug. 2007, pp. 1117-1124.

[51] Mahvash Mohammadi H., Langlois P. and Savaria Y., "A Pattern-Based Directional Interpolation Deinterlacing Algorithm," *IEEE Transactions on Consumer Electronics* (Jan. 2009, Submitted)

[52] Mahvash Mohammadi H., Savaria Y., and Langlois J.M. P., "Hybrid Video Deinterlacing Algorithm Using Reverse Motion Estimation" *IEEE Transactions on Circuits and Systems for Video Technology*, (Jan. 2009, Submitted)

Following are the detailed contributions in the HMC algorithm:

- A new motion estimation algorithm adapted to interlaced video data: The method uses forward and backward motion estimation within two previous and two next fields of the same and opposite parity to find the best possible match which has the same existing and missing lines as the original block.
- A new motion compensated technique to calculate missing pixels: The technique uses the amount of vertical motion within the reference fields to find out where the

missing lines of the block appear in the reference fields and which reference fields of the opposite or same parity filed should be used for deinterlacing.

- A new technique to detect the presence of fast and non-uniform motions: The proposed technique calculates the cross reference error between the block with half amount of the motion vector and the original block. This error is compared with the OMB error multiplied by a factor of α to recognize the presence of fast and non-uniform motions.
- A new edge-based method with a high angular resolution based on pattern comparison: It defines three-pixel pattern and uses a new method to calculate pattern correlation. It uses only nine pixels in each of two reference lines, but can recognize twenty-one edges in nine different directions.
- A new technique for pattern comparison: The technique is composed of a gradient-based criterion and the sum of absolute differences (SAD) to improve calculation of the pattern correlation.
- A new technique to recognize the edge direction with a low probability of being confused by misleading edges: The technique exempts comparison of some pairs of patterns to decrease the probability of the misleading edges occurrence.
- A hybrid structure based on motion vector reciprocity: The method switches between FFMC and PBDI based of on the reliability of the motion vector calculated by RME. FFMC is used to improve the vertical resolution when motion vectors are reliable and PBDI can prevent the artifacts of wrong motion vectors.

- A novel criterion to evaluate the reliability of motion vectors: It repeats motion estimation in a direction opposite to the calculated motion vector by replacing the original and reference fields to confirm validity of each motion vector. If the motion vector calculated by reverse motion estimation is the same as or is close to the primary motion vector, but in the opposite direction, the primary motion vector is assumed reliable.

Before we come up with our final hybrid method, we proposed a hybrid deinterlacing algorithm using motion compensation and enhanced ELA [53]. We also proposed optimization techniques to accelerate the enhanced ELA for the Xtensa reconfigurable processor [54] and implemented the PBDI algorithm in a high performance application-specific instruction-set processor (ASIP) [1].

[1] Aubertin P., Mahvash Mohammadi H., Savaria Y. and Langlois P., "A High Performance ASIP Implementation of the PBDI Intra-Field Deinterlacing Method," *IEEE NEWCAS 2009* (Accepted)

[53] Mahvash Mohammadi Hossein, Langlois J.M. P., and Savaria Yvon, "A Threshold-Based De-Interlacing Algorithm Using Motion Compensation and Directional Interpolation," *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Nice France, December 2006, pp. 459-462.

[54] Mahvash Mohammadi H., Savaria Y., and. Langlois P., "Real Time ELA De-Interlacing with the Xtensa Reconfigurable Processor," *The* 4th International *IEEE-NEWCAS Conference*, June 2006, p. 25-28.

The detailed contributions of these works are as follows:

1) A hybrid deinterlacing method based on OMB error: The method uses forward motion estimation within two previous reference fields using existing lines only to limit the propagation of errors. The amount of vertical motion within the reference fields is used to decide which reference fields of the opposite or same parity filed should be used to calculate missing pixels. The OMB error is compared with a threshold to recognize situations where the motion estimation is not accurate and switch to Enhanced ELA.
2) Optimization techniques to accelerate enhanced ELA on the Xtensa reconfigurable processor. The techniques are first based on low-level software optimizations to accelerate loops and arithmetic operations. Then new ELA-specific operations for the Xtensa reconfigurable processor in a specialized hardware structure are defined to accelerate the algorithm. The specialized hardware uses a parallel structure to calculate four missing pixels at a time. The combined software and hardware techniques result in a speed-up of 67× when compared to the base case.
3) An implementation of PBDI in an ASIP: The implementation focuses primarily on an efficient utilization of the available memory bandwidth. Custom instructions, application-specific registers and Very Long Instruction Words (VLIW) are used to create a virtual pipeline. This results in a speed-up of 1351 compared with a software implementation running on a general-purpose 32-bit RISC processor.

1.5. Thesis organization

This thesis is written in a paper-based format and contains copies of one published journal article in chapter 3, and of two submitted journal articles in Chapter 4 and 5.

Chapter 2 reviews the principles and classification of video deinterlacing algorithms and presents a review of related work. The chapter highlights key issues of deinterlacing algorithms in terms of efficiency and complexity.

Chapter 3 explains FFMC which is used in a hybrid deinterlacing algorithm as a primary method to achieve the best possible vertical resolution improvement. This work was published in IEEE Transaction on Consumer Electronics in August 2007.

In chapter 4, we put forward the PBDI algorithm. This method can be used in the hybrid method for situations in which motion vectors are not reliable. This work was submitted to IEEE Transactions on Consumer Electronics.

The hybrid deinterlacing algorithm and RME are proposed in chapter 5. The hybrid method uses RME to evaluate the reliability of the calculated motion vectors by FFMC. If motion vectors are reliable motion compensation by FFMC in used to deinterlace the sequences otherwise, PBDI is used. This work was submitted to IEEE Transactions of Circuits and Systems for Video Technology. Finally, the conclusion of the thesis is presented in chapter 6.

Chapter 2

Principles and Classification of Video Deinterlacing Methods

2.1. Theoretical context

According to the Nyquist–Shannon sampling theorem, any band limited and sampled signal can be exactly reconstructed if the sampling rate is more than twice the maximum frequency of the signal [25]. Interlacing is a down sampling procedure in which the vertical samples are halved (Figure 2.1). Therefore, the maximum spatial frequency of an image in interlaced scan is half of that in progressive scan. If the spatial vertical frequency of the image is less than half of the maximum spatial frequency in progressive scan, the missing lines can theoretically be reconstructed by deinterlacing filters. Even in progressive scan, some original video data with high spatial frequency may be discarded but, in deinterlacing, we only discuss the reconstruction of the missing data due to interlacing, and not because of a low sampling rate.

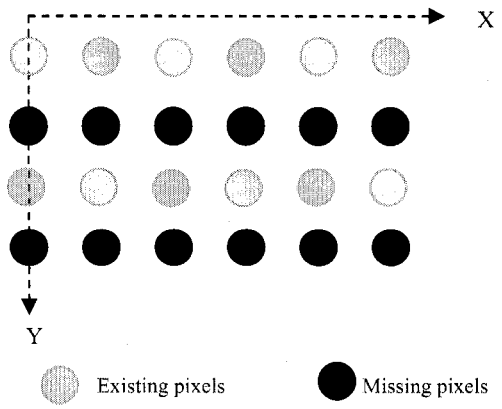


Figure 2.1. Down sampling in the spatial domain

Spatial frequency is measured by the number of cycles per unit of distance. In visual perception it is measured by number of cycles per degree. Although the subjective criterion of deinterlacing quality is a visual perception action, in order to remove the factor of distance between an observer and the screen, the spatial frequency is measured by the number of cycles per lines or screen.

Maximum spatial vertical frequency with 480 lines in progressive scan is 240 cycles per screen and in interlaced scan is 120 cycles/screen. Now, if the spatial vertical frequency of the image is less than 120 cycles per screen, the spatial interpolation can theoretically reconstruct the signal without any resolution loss otherwise it does not improve the vertical resolution. As there is no guaranty that spatial frequency of the original signal is less than the specified value, deinterlacing algorithms which use spatial interpolation techniques can not improve the vertical resolution.

Deinterlacing can also be viewed as a down sampling procedure in the time domain. Figure 2.2 shows the data of a video sequence in time domain. If data is represented progressively, each snapshot of a scene contains all samples of each position of the screen. Interlacing removes data of each pixel in every other frame in the time domain, and this corresponds to a down sampling action.

The maximum temporal frequency of each pixel in progressive scan is twice of that in interlaced scan. Since the time space between sequential fields in a video sequence is very small, the intensity of a pixel within two sequential fields is almost the same unless the pixel belongs to a moving object or the camera is moving. Due to this slow intensity change in static areas, the time frequency of each pixel in this area is very low and down

sampling applied by interlacing still satisfies the Nyquist rate. This is the basis of inter field deinterlacing methods in which the vertical resolution can be improved by temporal interpolation in static areas [64].

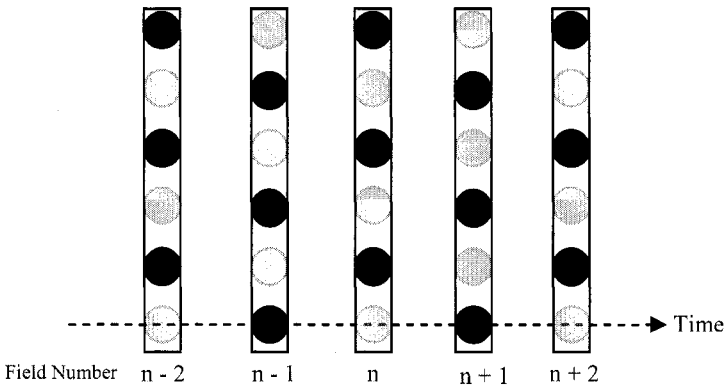


Figure 2.2. Down sampling in time domain

In situations where the Nyquist rate is not satisfied spatially or temporally, even if the best deinterlacing algorithm is applied, deinterlacing can not reconstruct the missing pixels reasonably well. Since video cameras are typically not equipped with optical pre-filters, aliasing effects happen in the deinterlaced output. Also, if pre-filtering was applied, severe blurring would result [68].

Video sequence data tend to be highly correlated. Two kinds of temporal and spatial correlations exist in the video data. Since the time space between sequential fields in the video sequences of conventional TV is 1/50 to 1/60 of a second, most parts of the scene are the same in two sequential fields. This kind of correlation is referred as the temporal correlation. The neighboring pixels in an image have a high probability of belonging to one object that has the same shape and texture. Therefore, the neighboring pixels likely

have the same intensity values. This kind of correlation is referred as the spatial correlation.

According to temporal and spatial correlation within the video data, all deinterlacing methods use some spatial and/or temporal interpolation techniques to calculate the missing lines. Spatial interpolation does not improve the vertical resolution but can remove some interlacing artifacts. The temporal interpolation can improve the vertical resolution in static areas but it may produce some artifacts in moving areas [15].

According to temporal or spatial interpolation, deinterlacing methods are generally categorized in four kinds of deinterlacing methods: intra field, inter field, motion adaptive, and motion compensated methods.

2.2. Intra field deinterlacing method

Intra field deinterlacing methods use spatial interpolation techniques to generate the missing lines. They calculate a missing pixel based on values of its neighboring pixels in the current field. According to the theory, intra field methods usually can not improve vertical resolution of interlaced sequences. This is especially true in horizontal edge areas in which the intensity toward the vertical directions is severely changed; this kind of deinterlacers gives poor image quality. On the other hand, due to the low calculation complexity and limited memory requirements, intra field deinterlacing methods are the most cost effective [8].

BOB (line averaging) and line repetition are the most cost efficient methods. In line averaging, each missing pixel is calculated as the average of its upper and lower pixels. It

causes blur and image resolution is unavoidably lost. Blur is an inherent drawback of averaging filters used by most intra field methods [47]. The line repetition method only repeats a previous line for each missing line. The display looks blocky and pixilated. The image resolution is not improved [22].

Edge-based deinterlacing methods produce good quality within the intra field methods. They attempt to recognize the direction of the edge at a missing pixel position and then interpolate it along that direction. Several edge-based algorithms have been proposed in the literature [14][40][34][10][40][75][20][46][56][7][11][29][26][37]. Edge-based line averaging (ELA) is the most familiar edge-based method [14]. It uses three neighboring pixels in each of the upper and lower lines to estimate the edge direction and then interpolate the missing pixel along that direction.

ELA works well in regions with dominant edges because the edge direction can be estimated accurately. In horizontal edges or areas with high spatial frequency, poor visual quality may result due to inaccurate estimation of the edge direction. Several algorithms have been proposed to improve the accuracy of ELA. In [10], additional measurements are introduced to improve the accuracy of the edge direction estimation. Enhanced ELA with median filtering is one of the most popular edge-based deinterlacing methods proposed in [46].

Misleading edge directions is another problem that decreases the effectiveness of edge-based methods [75]. Some edge-based methods use directional interpolation only for dominant edges [10][46] but the others consider that each missing pixel is located in an edge. However, there are some complex scenes for which an edge cannot be identified

for all missing pixel positions. Some edge-based methods use a median filter in the last stage of the deinterlacing process to prevent bursting pixels. Using a median filter calculates a value between the upper and lower pixels for a missing pixel which is not always true in the real scene.

Directional correlation dependent interpolation filter (DIF) is proposed in [40]. In [34], the pixels in horizontal edge regions are interpolated by the majority principle.

The Directional-Oriented Interpolation (DOI) method [75] performs a pattern-based comparison to detect the edge direction. It uses two upper and two lower reference lines to detect edges. DOI increases edge direction resolution and performs well in images with strong edges.

A novel intra deinterlacing (NID) is proposed in [29]. The method combines three edge based methods that leverage spatially local features. A Sobel filter is used to estimate the edge angle and local features.

Inpainting-based methods can also be categorized in intra field methods since they use the data of the current field [2]. Inpainting is a method to recover occluded area in an image by interpolation from their vicinity. The interpolation is based on elastica type energy of level lines structure.

2.3. Inter field deinterlacing methods

According to the temporal correlation between sequential fields, inter field methods use temporal interpolation techniques to calculate the missing pixels. These methods improve the resolution of the image in static areas but they produce the combing artifact

which is very distracting in motion areas. Also, a memory buffer is needed to save the fields used for interpolation. Combing artifacts are created through the horizontal displacement of an object in every two sequential fields. When there is no motion in the picture, the best value for a missing pixel is the value of a pixel in the same position in the previous or next field.

Field combining or weave is the simplest intra field method. In this technique, each pair of two consecutive fields is merged together to form a frame [3]. This generally improves image resolution and works well if there is no motion. Very few deinterlacers use this kind of deinterlacing as a primary algorithm due to the introduction of combing effect in moving areas.

Inter field averaging uses temporal averaging. The missing pixels are calculated as the average of pixels in the same position in the previous and next fields [4]. In static areas this method works well but in moving areas it causes combing artifacts.

Vertical-temporal median filtering calculates the median of the upper and lower pixels in the current field and the pixel in the same position in the previous field [4].

2.4. Motion adaptive deinterlacing

Many motion adaptive methods are reported in the literature [46][47][18][17][30] [42][9][77][59][76][66][45]. These methods combine the advantages of intra field and inter field algorithms. They usually use spatial interpolation in motion areas and temporal interpolation in static areas. A motion detection algorithm is used to distinguish motion and static areas and it is the most important component of the motion adaptive methods.

Some motion adaptive methods consider a boundary area in which the motion detection is not reliable [18][30][59]. A combination of the spatial and temporal interpolation values is used in the boundary areas. A memory buffer is required to save neighboring fields which are used for motion detection and temporal interpolation.

Different motion detection algorithms have been proposed in the literature [32][43][37][39]. They calculate the brightness differences of some neighboring pixels in the neighboring fields and compare them with a threshold to indicate the presence or absence of a motion. In the Two-field motion detection method, the missing pixel is calculated as the average of the upper and lower pixels and is then compared with the pixel in previous field at the missing pixel position [32]. Three-field motion detection method compares the difference of pixels in the previous and next fields at the missing pixel position with a threshold value. Four-field and five-field motion detection algorithms have also been proposed in [43] and [37], respectively.

Same parity fields or sequential fields are used in motion detectors. Motion detection algorithms using successive fields need to interpolate the missing pixels. Inaccurately interpolated data can cause incorrect decisions that can propagate to the next fields. Using same parity fields requires more memory buffers. It also increases the temporal distance between missing pixels and the pixels used for motion detection which can cause incorrect decisions.

"Motion missing" and "false motion" are two major problems of motion detection algorithms. Motion missing happens when there is a motion but the motion detector does not detect it. In this case, the area is considered static and temporal interpolation is used.

This causes image quality deterioration. False motion happens when there is no motion but the motion detector erroneously reports a motion. In this case, spatial interpolation is used instead of temporal interpolation. False motion does not generate severe artifacts but the vertical resolution of the image is not improved [18].

D. Haan et al., in [18] and [17], proposed a motion adaptive method in which the concept of brightness profile pattern difference is introduced for acquiring noise immune characteristics for motion detection algorithm. This method uses the brightness differences of adjacent pixels in four successive fields to calculate a brightness profile pattern difference. Then, a 3×3 Sobel operator is used for coarse edge detection in three orientations (0, -45°, +45°). Finally, a soft switch calculates the final value for a missing pixel according to the normalized motion detection, spatial and temporal interpolated values.

The method proposed in [47] uses directional interpolation by ELA with median filtering and a 4-field horizontal motion detection to prevent missing motion in the fast motion area. The motion detector uses difference between the forward-forward and current fields as well as forward and backward fields. Then results of these two differences are ORed together. A temporal prediction is also done by the motion detector to estimate the amount of horizontal motion between the forward and backward fields. The weave method is used in static areas and ELA with median filtering is used in motion areas.

Y. Kim et al. proposed a recursive motion detection algorithm in [30]. This algorithm uses a history of overall image motion detection values in the previous field

and results of motion detection in the current field to calculate the final decision value. This value is used as the mixing ratio between temporal and spatial interpolation. Linear averaging is used as the spatial interpolation and Weave for temporal interpolation.

Sung-Gyu L. and Dong-Ho L. proposed a motion-adaptive deinterlacing algorithm based on an edge-based median filter (EMF) and adaptive minimum pixel difference filter (AMPDF) [42]. This method classifies the area into moving, background and boundary areas by using different threshold values. Then, AMPDF is used to compensate for the missing-motion error, which is an important factor in motion adaptive methods. The EMF is used to efficiently interpolate moving diagonal edges, with a 5-point median filter using edge information.

2.5. Motion compensated deinterlacing methods

Motion compensated methods are the most powerful and complex deinterlacing methods. These methods try to virtually remove motion from video sequences and then use the benefit of temporal interpolation in static and motion areas [22]. Motion estimation is used to calculate the displacement of blocks or objects within the neighboring fields. The efficiency of motion compensation methods highly depends on the accuracy of motion estimation [27].

Motion estimation methods calculate the displacements of moving pixels, blocks or objects. In traditional motion estimation algorithms, each field is partitioned into blocks, usually of size 8×8 or 16×16. Then, the motion estimator will search the neighboring fields to find the most similar block to each block of the current field called "optimal

matching block" (OMB) (see Figure 2.3). The displacement between a block and its OMB is called a motion vector. The search method and criteria for matching blocks are two important factors which affect motion estimation.

Conventional motion estimation algorithms use block matching to find an OMB. The algorithm calculates the mean absolute difference (MAD) or the sum of absolute differences (SAD) between a block in the current field and blocks in the neighboring fields within a search window. Phase correlation can also be used to find the best matching block. It works based on the Fourier transform and can provide better results than the conventional algorithms but involves a much greater computational complexity.

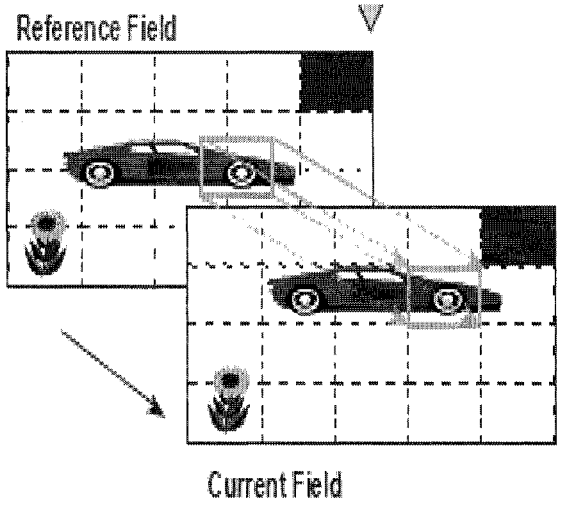


Figure 2.3. Block matching algorithm [8]

Different search methods have been proposed in the literature [3][63][61][31][24] [16][44][48][78][49]. In the following, details of some important search methods are presented.

The full search method provides the best results but requires the highest computational complexity that is difficult to implement in the real time systems. In this method, all possible blocks in the search window are compared to find the OMB.

Other methods are faster than the full search method. They decrease the number of compared blocks in the search window but they work based on the assumption that the MAD monotonously increases with the distance between the matching block and the OMB. This assumption is not always true, and the search can be trapped in local minima. Cross search and Tree Step search are two popular methods within this kind of algorithms.

In the cross search method, only five blocks in different positions of the search window (cross shaped) shown in Figure 2.4 are selected for comparison during the first step. If the central block has minimum MAD then the process is repeated for four other blocks with half of the previous distance from the center block. If one of other four blocks has the minimum MAD then the center of a new cross will be placed on that point and the algorithm is again repeated [16].

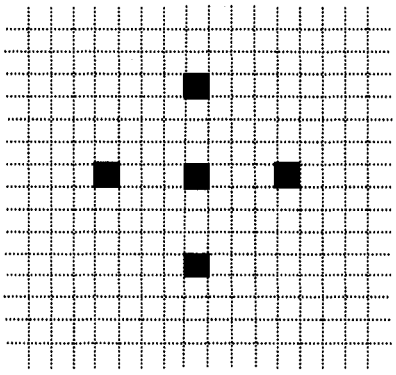


Figure 2.4. Cross Search Pattern

Three Step Search (TSS) is another fast search method. This method is like cross search but with a different pattern shown in Figure 2.5. In the first step, all nine blocks are compared to find a block with minimum MAD. If the central block has minimum MAD, the algorithm is repeated in the same place but with half distance between blocks. If any other block has a lower MAD, the pattern is moved to that point and the algorithm is again repeated while the distance between the blocks is halved. [31].

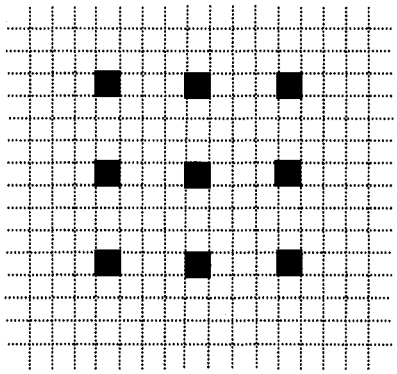


Figure 2.5. Three Step Search Pattern

The 2-D Logarithmic Search (TDL) [24], Conjugate Directional Search (CDS) [63], Orthogonal Search Algorithm(OSA) [61], Diamond Search (DS) [78], New Three Step Search (NTSS) [44], Block-Based Gradient Descent Search (BBGDS) [48], Adaptive Cross Search (ACS) [49], Dual-Cross Search (DCS) [67], Adaptive Dual-Cross Search (ADCS) [3], and One at a Time Search (OTS) are other search methods proposed in the literature.

Forward and backward motion estimation algorithms search for OMBs in the previous or following fields, respectively. Bidirectional motion estimation uses both directions, and it solves the problem of appearing and disappearing parts of the scene covered and uncovered by other objects.

Interlaced video data complicates motion estimation and compensation. Motion estimation may not be accurate for interlaced data because of the alteration of existing lines between odd and even fields. Most algorithms use pre-filtering with an intra field method to calculate missing lines prior to motion estimation. Motion estimation then depends on the interpolated pixels as well as the original pixels. Thus, any interpolation error may be propagated into the subsequent output frames. This is a common drawback of the motion compensated methods using pre-filtering. Some approaches, such as the five-point median filtering, are used to prevent error propagation [27]. Although this is a very effective method, the median filter can introduce aliasing in the deinterlaced image [22]. Some methods use only same parity fields for motion estimation in which pre-filtering is not necessary.

Motion compensation may generate artifacts if the motion vectors are not correct. There are some challenging circumstances in which conventional motion estimation algorithms can not provide reliable motion vectors. Conventional motion estimators may fail in video sequences with rotation or scaling, when there are appearing and disappearing regions, if an object moves beyond the search area, or when there is sub-pixel motion. Fast search methods can also fail to calculate accurate motion vectors. Even if the motion estimator successfully calculates the motion vector, how to calculate the missing lines remains an important issue in interlaced video data [49].

Different kinds of motion compensated deinterlacing algorithms have been proposed [64][15][8][10][43][27][69][72][36][55][62][58][21][33][13][17][67][43][65] [28][35][5][56][74][6][70][73][23][79]. In the simplest conventional motion

compensated deinterlacing algorithm, pre-filtering by line averaging is applied in the current field. Then, the motion estimator finds the OMB in the previous field. The missing lines of the blocks in the current field are replaced by the corresponding lines of the OMB in the previous field.

Some motion compensated methods combine elements from two or more classes. Several method use motion compensation only if motion vectors are reliable and an intra field deinterlacing algorithm is used in case of unreliable motion vectors. A measure of motion vector reliability is needed to qualify the motion vectors. These methods exploit the advantages of motion compensation to improve vertical resolution and prevent artifacts in case of incorrect motion vectors. We now review some complex motion compensated deinterlacing algorithms.

In [27], a motion compensated algorithm is proposed in which motion estimation is applied between the same parity fields n-1 and n+1 to calculate the motion vectors. Due to the use of same parity fields, this method does not need pre-filtering for motion estimation. Then the motion vectors are refined to produce a new field in between n-1 and n+1 with half motion vectors. This field is used to calculate the missing lines. For further improvement, a five point median filter with temporal emphasis is used to reduce the interpolation error caused by incorrect motion.

A four-field motion compensated method is proposed in [6]. Matching blocks in the same parity fields n and n-2 are used for motion estimation but the criterion to select the best matching block is different from conventional methods. SAD1 is calculated by matching blocks from the current field to field n-2. Half of the motion vector corresponds

to SAD1 in forward and backward directions and refer to two blocks in fields n-1 and n+1 respectively. SAD2 is calculated from the error between these two blocks. The final motion vector is calculated based on the matching blocks that minimize SAD1 + SAD2. The motion vector, the SAD and block diverse amount are used to switch between motion compensation and an intra field method. Block diverse amount counts the number of pixels for which the differences between the motion compensated fields are larger than a given threshold.

A complex hybrid motion compensated method is proposed in [8]. The method switches between four deinterlacing algorithms: weave, an edge-based method, and local and global motion compensation based on amounts of SAD and IPDC (interlaced pixel distortion classification). The method uses the same motion estimation proposed in [6]. Global motion estimation is also applied to support four global motions of translation, isotropic, affine, and perspective models.

D. Wang et al proposed a hybrid motion compensated algorithm in [70]. The method combines motion compensation with a vertical-temporal filter based on reliability of motion vectors. Motion vector reliability is measured using the a posteriori probability of motion vectors.

A hybrid motion compensated method described in [23], uses five successive fields for forward and backward motion estimation. The same and opposite parity fields are used to calculate four motion vectors. A probabilistic criterion is used to select the most reliable motion vector and switch between four kinds of deinterlacing.

An adaptive deinterlacing method combining four different approaches is proposed in [33]. First, they combine the results of linear vertical interpolation and inter field averaging. The weight of each method is calculated according to the directional difference of neighboring pixels in the spatial and temporal domains. The third method uses directional interpolation by steerable filters. The last method performs temporal interpolation by motion compensation. The motion estimator uses both forward and backward motion estimation. If the motion estimator results are not accurate enough, the algorithm will not give adequate weight to the motion compensation part.

In [78], line averaging is used prior to motion estimation to calculate the missing lines. Then, the motion estimator calculates the motion vector between the current and previous or next fields. By combining information from the current field, the previous same parity field, and the motion information, one of the five modes is selected: stationary, forward prediction, backward prediction, bidirectional prediction or intra-field. Finally, the spatio-temporal interpolation filter, adaptively controlled by the motion and deinterlacing mode information, generates the missing pixels.

In [55], four kinds of deinterlacing methods are combined. Vertical averaging and directional interpolation based on steerable filter are two spatial interpolations techniques used in this method. Inter field averaging and motion compensated interpolations are two temporal interpolations methods combined in this method. The weight of each method is calculated based on the likelihood of the corresponding filters yielding the most appropriate interpolation result.

A motion adaptive compensated method is proposed in [64]. The method uses motion compensated inter-field interpolation for normal areas. Intra field interpolation is used for areas with high motion or shape changes. The method uses bidirectional motion estimation using small blocks of 4 × 4. An area of 5 × 5 blocks is used to check the smoothness and reliability of the motion vectors.

R. Li et al. proposed a method in which a phase correction filter is used prior to motion estimation. The filter corrects the phase difference between the opposite parity fields caused by the alternating lattice between them. Then an intra field or motion compensation algorithm is applied based on the amount of OMB error [43].

MVBOB is a hybrid method that combines different kinds of deinterlacing algorithms [79]. A motion adaptive method is used to calculate missing pixels prior to motion estimation. The weave method is used in static areas and different intra field methods such as line averaging, ELA and EDI can be used in motion areas. The motion detection algorithm uses 2 to 16 fields to discriminate between static and motion areas. Then forward or backward motion estimation is used to calculate motion vectors. Different search method such as One-Time-Search, N-Step-Search, Diamond search and full search can be used for motion estimation. When motion vectors are calculated, OMB error is used to recognize inaccurate motion vectors. Block-based or pixel-based motion compensation is finally used to calculate missing pixels.

2.6. Deinterlacing algorithms comparison

Table 2.1 summarizes the issues relevant to the different deinterlacing methods with a qualitative comparison. Intra field methods have the lowest computational complexity but also the poorest image quality. Inter field methods work great in still areas but provide the most annoying artifacts in moving areas. These methods have low computational complexity and their memory requirement is up to one or two fields depending on the temporal interpolation technique used. The quality of motion adaptive methods depends on motion detection algorithms. The best quality can be achieved in still areas but the vertical resolution in moving areas is not improved. The memory requirements of these methods depend on the number of fields used for motion detection.

TABLE 2.1. DEINTERLACING METHODS COMPARISON

Deinterlacing method	Efficiency (quality)	Performance (computational complexity)	Memory requirement
Intra field	No resolution improvement, flickering, blur	Least	Up to a few lines
Inter field	Combing artifact in moving area, Best quality in still areas	Medium	One field (minimum)
Motion adaptive	Good quality (Poor quality in moving area)	High	From one to a few fields
Motion compensated	Very good quality (Highly depend on the accuracy of motion vectors)	Very high	From one to a few fields
Hybrid Motion compensated	Best Quality	Highest	From one to a few fields

Motion compensated methods usually provide the best quality. The effectiveness of these methods highly depends on the accuracy of motion estimation which is the most

complex part and which determines memory requirements. However, perfect motion estimation is not practical in all situations and these methods may produce artifacts in case of inaccurate motion vectors.

Hybrid motion compensated methods try to provide the best quality with no artifacts. A measure of motion vector reliability is a key issue in these methods to achieve the maximum vertical resolution improvement with no artifacts. These methods have the highest computational complexity. The memory requirement for these methods depends on the number of fields used for motion estimation and motion vector qualification.

2.7. Image qualification problem

Benchmark progressive sequences are often used to evaluate the performance of deinterlacing algorithms. The sequences are first interlaced and then deinterlaced by the algorithms. The difference between the original and deinterlaced sequences is used for objective and subjective quality assessments.

For subjective qualification, the images or sequences are shown to people who are then asked to evaluate them. Subjective qualification is often too time consuming and also human factors such as culture and age can affect the perceived quality.

Objective evaluation is fast and practical but current objective metrics are not fully correlated with perceived quality measurements. Traditional objective metrics are error-sensitivity based. They evaluate the deinterlacing methods based on the error between the original and deinterlaced sequences. Mean square error (MSE) and peak signal to noise ratio (PSNR) are the most popular error-based methods. However, these metrics are often

criticized because they are not well correlated with the human visual system. The human visual system extracts structural information from the viewed picture and it is not an error based system [71]. It is more sensitive to blur than to other kinds of noise but error base metrics are not. Therefore, a measurement of structural information loss can provide a good approximation to perceived image distortion. A great deal of effort has been made in recent years to develop objective image quality metrics that correlate well with perceived quality measurements. Unfortunately, only limited success has been achieved and defining a good, objective image quality metric remains an open problem.

Chapter 3

A Five-Field Motion Compensated Deinterlacing Method Based on Vertical Motion

Hossein Mahvash Mohammadi, Pierre Langlois and Yvon Savaria

Abstract *— A new motion compensated deinterlacing method using forward and backward motion estimation is proposed in this paper. Bi-directional motion estimation is performed using two previous and two subsequent fields. The motion estimator uses pre-filtering prior to motion estimation for the current and the subsequent two fields. The motion estimator finds a single optimal matching block in the same or opposite parity reference fields. Motion compensation is done according to the amount of vertical motion within the reference fields to achieve the highest vertical resolution improvement. A novel technique to prevent the appearance of visual artifacts in the presence of fast-moving objects is proposed. Experimental results show that the proposed method performs better than the conventional deinterlacing methods, based on objective and subjective criteria.*

Index Terms — Deinterlacing, Motion Compensation, Motion Estimation, Bi-directional Motion Estimation.

3.1. Introduction

Interlacing was originally incorporated into video signals because it allows the transmission of a subjective amount of information deemed acceptable with less

bandwidth. Although interlacing is not the best technical solution to reduce transmission bandwidth, compatibility with existing standards is maintained in the TV broadcast market [1]. Interlacing reduces the image quality through the introduction of artifacts in the image and it complicates many image-processing tasks, particularly scanning-format conversions [2].

Deinterlacing and scan up-conversion are inevitable in order to support the existing variety of interlaced standard source formats adopted in HDTV. Depending on temporal and spatial correlation within the video data, deinterlacing methods use spatial and/or temporal interpolation techniques to calculate the missing lines. These methods can increase the vertical resolution of interlaced video signals and remove interlacing artifacts.

Deinterlacing methods can be classified based on whether they exploit spatial or temporal interpolation techniques. Intra-field deinterlacing methods only use spatial interpolation to calculate missing lines. Line repetition and line averaging are the two simplest intra-field methods. Edge-based deinterlacing methods have the best quality within the intra-field category. They interpolate missing pixels toward the highest directional correlation. The edge-based line average (ELA) algorithm is widely used because it involves simple calculations and generally provides satisfactory results [3].

Inter-field deinterlacing algorithms implement temporal interpolation techniques. They provide the best quality in still areas but can produce combing artifacts in moving areas. Motion adaptive deinterlacing methods combine the advantages of intra-field and inter-field deinterlacers. They use spatial interpolation in motion areas and temporal

interpolation in static areas. Thus, an accurate motion detection algorithm is needed to distinguish between motion and static areas.

Motion compensated methods are the most powerful but also the most complex deinterlacing methods. The motion compensation technique was originally conceived to remove the temporal redundancy of video sequences in video coding. Although applying this technique is complicated for interlaced sequences, it can virtually remove motion from video sequences and then use the benefit of temporal interpolation in motion areas as well as static areas [1]. Motion compensated methods are highly dependent on the accuracy of the motion estimation, and this is one of the most challenging issues in these methods [4].

This paper is organized as follows. In section 3.2, an overview of motion compensated deinterlacing methods is presented. Section 3.3 presents the proposed deinterlacing algorithm. Section 3.4 shows experimental results and a comparison with previous methods. Section 3.5 concludes the paper.

3.2. Overview

3.2.1. Theoretical Discussion

Interlacing can be considered as a down sampling procedure in which the vertical sampling rate is halved. If the spatial vertical frequency of the original image is less than half of the maximum spatial frequency corresponding to progressive scan, then the Nyquist criterion is satisfied and the interlaced signal can theoretically be reconstructed by intra-field deinterlacing. As this condition is not always satisfied, intra-field methods

do not produce satisfactory results in images with high vertical frequency details and usually produce flickering artifacts in these images.

Alternatively, deinterlacing can be viewed as a down-sampling procedure in the time domain. If the temporal frequency of each pixel is less than half of the maximum frequency of the pixel corresponding to progressive scan, temporal interpolation can reconstruct the missing pixels perfectly. Over short time periods, the light intensity of objects in the real world is almost constant. Hence, the intensity of each pixel in the screen is also constant, unless objects in the scene or the camera are moving. Therefore, the temporal frequency of the pixels in static areas is very low and down sampling applied by interlacing still satisfies the Nyquist criterion. Thus, missing pixels can be reconstructed by temporal filters. This is the theoretical basis for inter-field deinterlacers, in which the vertical resolution can be improved by temporal interpolation in static areas [5].

3.2.2. Motion Estimation and Compensation

Motion compensated methods use motion estimation algorithms to calculate the motion vectors for each block in the current field. Motion estimation is the most important and complex part of a motion compensated method. The effectiveness of motion compensation is highly dependent on the accuracy of the motion vectors [6]. Motion vectors are calculated by the displacement between a block in the current field and the most similar block called the "optimal matching block" (OMB) in the neighboring fields.

Conventional motion estimators use a block matching algorithm to find the OMBs. The block matching algorithm calculates the Mean Absolute Difference (MAD) between each block in the current field and candidate blocks in the neighboring fields within a search window. Among the different search methods, the Full Search method provides the best results but it involves the greatest computational complexity. In this method, all possible blocks in the search window are compared to find the OMB. The computational complexity is directly proportional to the area of the search window.

Other methods are faster than the full search method [7]-[16]. These methods decrease the number of compared blocks in the search window, but they are based on the assumption that the MAD monotonously increases with the distance between the matching block and the OMB. This assumption is not always true, and the search can become trapped in local minima that do not correspond to an OMB, so these methods may lead to suboptimal solutions.

Alternation of existing lines in interlaced video sequences complicates motion estimation and motion compensation. Different solutions have been proposed to overcome this problem. Some motion compensation methods use pre-filtering to calculate missing lines prior to the motion estimation and they use pre-deinterlaced pixels for motion estimation. However, these techniques propagate the estimation error to the next fields. Other methods perform motion estimation on same parity fields only. These methods assume uniform motion within the reference fields. They may produce artifacts for non-uniform or very fast motion when a moving object is present in the opposite parity field but is not present in the search window in the same parity fields.

If the missing lines of the current field are again missing in the reference fields, motion compensation is usually unable to increase the vertical resolution. This results in error propagation, which is a common drawback of all recursive methods.

Unidirectional or bidirectional motion estimation can be used by motion compensated methods [4][6][17]-[27]. In unidirectional motion estimation, forward motion estimation alone is used in the previous frames that have already been deinterlaced. In contrast, bidirectional motion estimators use previous and subsequent frames or fields for forward and backward estimation respectively. In backward estimation, missing pixels of subsequent fields have not yet been calculated, so these methods either use pre-filtering or only use the same parity fields for the motion estimation. Motion estimation in the same parity fields can be performed without pre-filtering because the existing lines have the same vertical position. Bidirectional motion estimation can solve the problem of appearing and disappearing parts of the scene when they are covered and uncovered by other objects.

In the simplest traditional motion compensated method, pre-filtering by line averaging is used to calculate the missing lines of the current field. Then the motion estimator searches the previous frame to find the OMB, calculates the motion vectors and generates the missing lines by temporal interpolation along the motion vectors. In other words, the missing lines are replaced by the corresponding lines of the OMB in the previous field as follows:

$$F_o(x,y,n) = \begin{cases} F_i(x,y,n), & y \bmod 2 = n \bmod 2 \\ F_o(x-d_x, y-d_y, n-1), & otherwise \end{cases} \tag{3.1}$$

where F_i is the input field, F_o is the output frame, d_x and d_y are the horizontal and the vertical motion vectors and n refers to the frame number.

3.3. Proposed deinterlacing algorithm

3.3.1. Motion Estimation

As already mentioned, motion compensation techniques are highly dependent on the accuracy of motion estimation. Unfortunately, there are challenging circumstances in which even the best motion estimation methods may fail to calculate reliable motion vectors. These include situations where there is sub-pixel motion, rotation or scaling in the video, movement of an object beyond the search window and masking between objects.

In [17] we proposed a motion compensated method in which two previous fields are used for forward motion estimation. Only original lines were used for unidirectional motion estimation of same and opposite parity fields and no pre-filtering was applied prior to motion estimation. The motion compensation was based on the amount of vertical motion within the reference fields.

In the method proposed in this paper, the two previous fields and the two subsequent fields are used as reference fields for the forward and backward motion estimation. Pre-filtering is used by line averaging in the current and the subsequent fields. The Full Search method is used for block matching to find an OMB in each reference field. Then

the OMB with the smallest Sum of Absolute Difference (SAD) is selected as the final OMB to calculate the motion vectors.

The OMB error is defined by the SAD between a block and its OMB as follows:

$$Error(OMB) = \sum_{x=1}^{Xblksize} \sum_{y=1}^{Yblksize} \left| F_i(x, y, n) - F_i(x + d_x, y + d_y, m) \right| \qquad (3.2)$$

where m refers to the field in which the OMB is found. Although the motion estimation is bidirectional, only one OMB is selected from either the forward fields or the backward fields without regards to parity. Using opposite and same parity reference fields in the forward and backward direction helps to find the OMB more accurately.

3.3.2. Motion Compensation

The proposed motion compensation technique is based on the amount of vertical motion within the reference fields. To improve the vertical resolution of the images, the proposed method checks whether the missing lines of a block are present in the neighboring fields. If the missing lines are not present in the neighboring fields, pre-deinterlaced or pre-filtered pixels must be used. For same parity current field and reference field, the existing lines of a block in the current field and the matching block in the reference field have the same parity line number. For different parity current and reference fields, existing lines in the current field have one pixel displacement from corresponding blocks in the reference field.

If the vertical motion of an object in two sequential fields is an odd number of lines, the existing lines of the object in the current field do not change in all reference fields. This situation is shown in Figure 3.1.a, in which the existing lines of the capital letter "A"

are the same in all reference fields. Therefore, the OMB can be found in any reference field because the existing pixels of the blocks are the same. In this situation, the vertical resolution cannot be improved because pre-deinterlaced or pre-filtered lines are used for the missing lines. Only pre-deinterlaced lines that have already been obtained from existing lines can improve the vertical resolution.

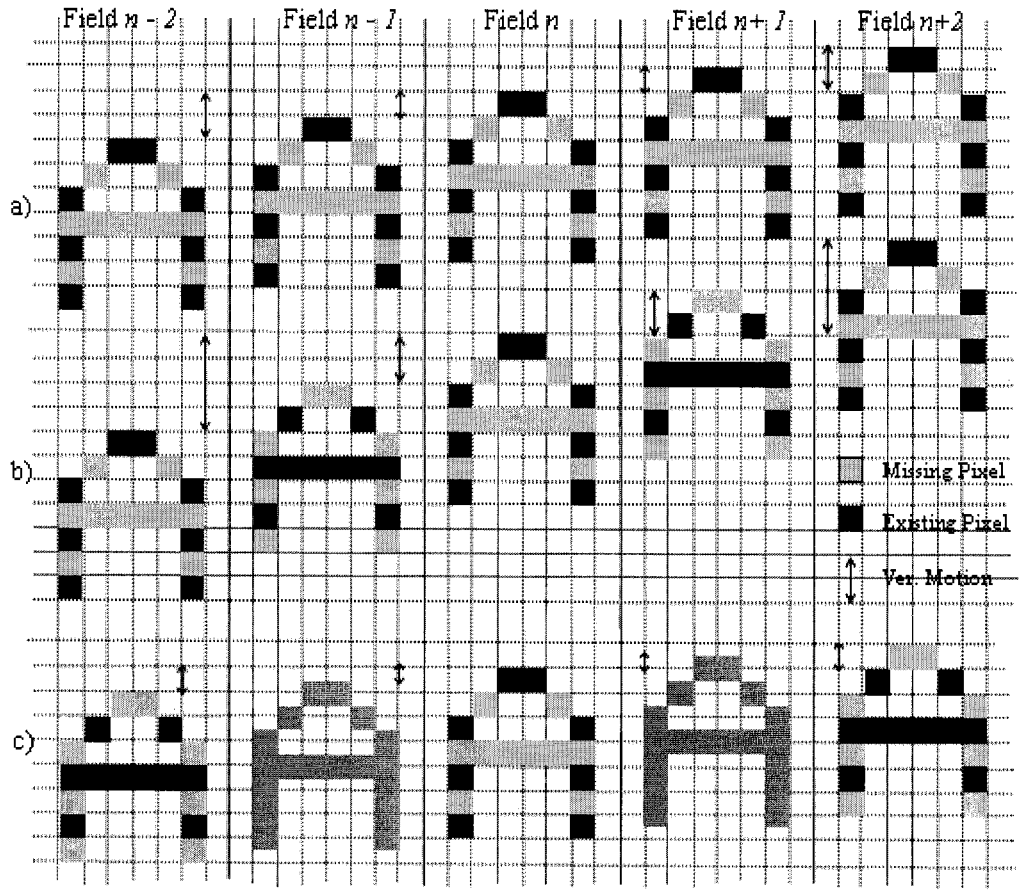


Figure 3.1. Block Motion a) Odd number of lines for vertical motion b) Even number of lines for vertical motion c) Half pixel vertical motion

If the vertical motion of an object is an even number of lines between two sequential fields, the existing lines of the block in the current field correspond to the missing lines in

the immediate next or previous field. They again correspond to the same lines in the previous and next same parity fields. Therefore, in this case, the OMB should be found in the previous or next same parity fields because of matching pixels in the corresponding fields. This is shown in Figure 3.1.b.

In the presence of sub-pixel movement as shown in Fig 3.1.c, if there is only a half pixel vertical movement per field, the total motion between the current field and the next or previous same parity fields will be one pixel, but the existing lines of the current field will be missing. Further reference fields are needed so that the existing lines of the current field may reappear and the OMB may be found accurately. This increases the computational complexity. However, if the OMB is found in the previous or next same parity fields, the missing lines of the block are present in the OMB and they can be used to improve the vertical resolution.

When the OMB is found, the motion estimator returns the OMB error calculated as in (3.2), the motion vectors and the index of the reference field in which the OMB was found. The motion vectors are calculated by the displacement of the block and its OMB as follows:

$$MV = \left(d_x, d_y, m\right) \qquad (3.3)$$

where d_x and d_y are the horizontal and vertical displacements between a block and its OMB and m refers to the reference field in which the OMB is found.

Figure 3.2 shows the flowchart of the proposed method. If the OMB is found in the same parity field with an odd number of lines of vertical movement per field, or in an opposite parity field, the missing lines of the block in the current field are replaced with

the corresponding lines of the OMB. In this case the motion vectors are more reliable because the OMB and its corresponding block have the same missing and existing lines, but the motion compensation calculates missing lines from pre-deinterlaced or pre-filtered lines because they are not present in a reference field.

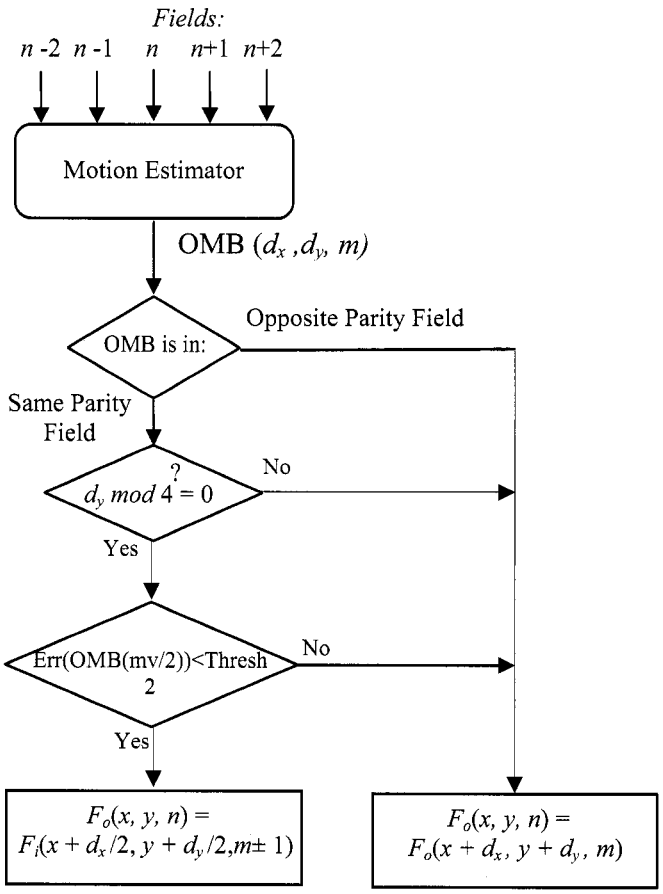


Figure 3.2. Flowchart of proposed method

If the OMB is found in a previous or next same parity field and the amount of vertical motion per field is an even number, the amount of vertical motion between the current field and the reference field in which the OMB is found is a number multiplied by four. In this case, the missing lines of the block in the current field are replaced by the existing

lines of the block with half of the motion vectors in the previous or next parity field, respectively. This is the best situation since it is possible to improve the vertical resolution. As shown in Figure 3.1.b, in this case the existing and missing lines for a block in the current field and the OMB are the same. This allows for a more accurate calculation of the OMB. Furthermore, the missing lines exist in the opposite parity fields with half vertical motion and this improves the vertical resolution.

3.3.3. Preventing artifacts caused by fast moving objects

Using the same parity fields for motion estimation may produce artifacts in situations with fast or non-uniform moving objects. Figure 3.3 shows three sequential frames of the "Table tennis" sequence. The background in these sequences is static and the tennis ball is moving fast. The frames n and n+2 have the same background data in the specified blocks, and the OMB for the block in frame n is found in the same position in frame n+2.

Now if the block with half motion in frame n+1 is used for motion compensation, an artifact will be introduced. This is because the tennis ball does not appear in frames n and n+2 of the block of interest, but it is present in this block for frame n+1. Figure 3.3.d illustrates the resulting motion compensation artifact for frame n.

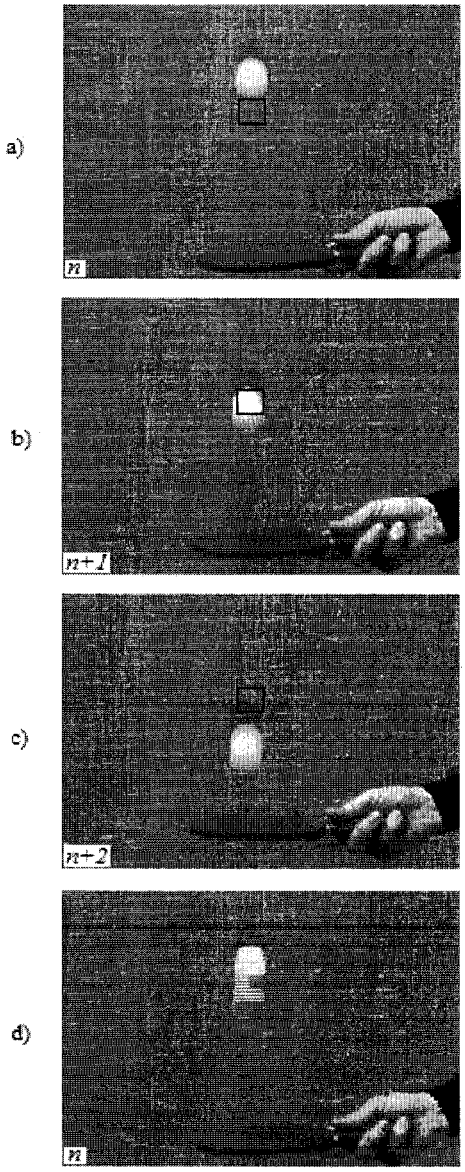


Figure 3.3. The artifact of same parity motion estimation

The same kind of artifact can be produced when there is a non-uniform motion. Figure 3.4 shows this situation in which the vertical movement of the ball is not uniform. The ball is going down in the first two frames but is going up in the next frame. As shown in Figure 3.4.a and 3.4.c, the OMB for the block containing the tennis ball in frame n is found in the next same parity frame n+2. Now if the block with half motion

vector in frame n+1 is used for motion compensation, the artifact shown in Figure 3.4.d. is observed in the deinterlaced frame n because the block contains background pixels.

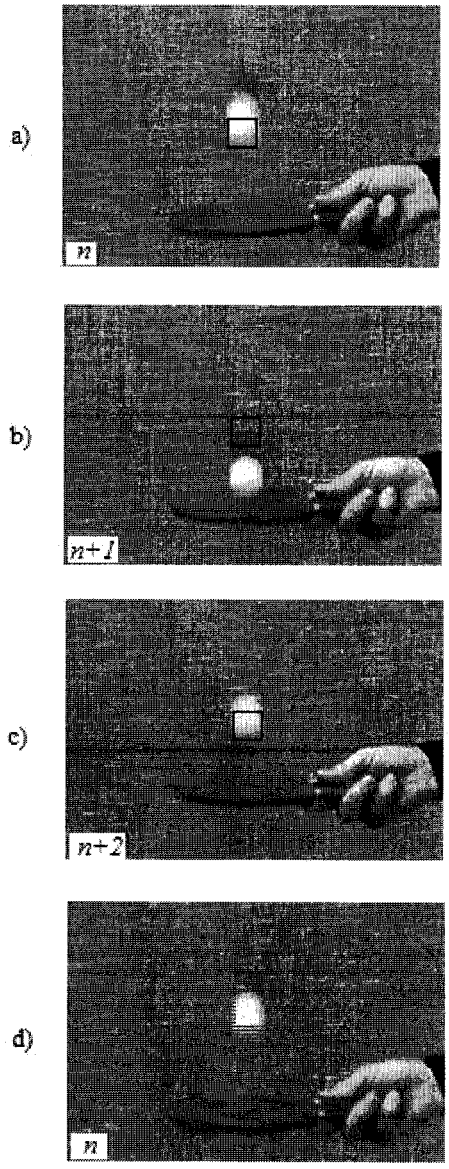


Figure 3.4. Artifact caused by non-uniform motion

To remove this type of artifact, the proposed method uses an extra check before the block with half motion vector is used for motion compensation. The verification involves

calculating the sum of absolute differences between the current field and the block with half motion in the opposite parity field. If the error is smaller than a certain threshold, the block with half motion vector in the opposite parity field is matched with the block in the current field and it is used for motion compensation; otherwise, the missing lines are replaced by the pre-filtered or pre-deinterlaced lines of the OMB.

The threshold is selected according to the amount of OMB error. The OMB error shows how much the OMB and its related block are correlated. This correlation must be kept when the block with half motion vectors is used for motion compensation. However, because the block with the half motion vectors is in the opposite parity field, the existing and missing lines of this block are not similar to the block in the current field. Therefore, the amount of error between the block in the current field and the block with half motion vector may be greater than the OMB error. Thus the threshold can be calculated by a coefficient multiplied by the OMB error as follows:

$$Threshold = \alpha \ . \ Error(OMB) \qquad (3.4)$$

where α is a coefficient calculated empirically and *Error(OMB)* is calculated by (3.2). Different values of the α coefficient were tested to find an optimal value. Overestimating α may cause the type of artifacts discussed here and underestimating it increases the use of pre-deinterlaced or pre-filtered lines, preventing an improvement of vertical resolution.

3.4. Performance results and analysis

Several progressive color sequences were used to evaluate the proposed method. The sequences were first interlaced, and then deinterlaced by several methods. The mean square error between the deinterlaced and original sequences was calculated. To decrease the computational complexity, the proposed motion estimation is performed only for the luminance component "Y", but its results are used for motion compensation in all the three RGB color components. Final results are obtained from the average of the MSE in all three color components.

A block size of 8 × 8, a search window size of 32 × 32, and the full search method were used for the proposed method and the conventional motion compensated method. Line averaging was used for pre-filtering for the current field in the conventional motion compensated method and for the current field and the next two fields in the proposed method.

Due to a lack of detail or unavailability of the implementation code of existing deinterlacing methods in the literature, the performance of the proposed algorithm is compared with that of well-known methods. Table 3.1 shows the PSNR over 50 frames for different deinterlacing methods. According to the PSNR, the new method performs better than the other methods in almost all sequences. However, the combing artifact, typical of the weave method, and the flickering artifact, typical of intra field methods in areas with high spatial frequency, makes them poor choices, even if they paradoxically exhibit a higher PSNR in some sequences.

TABLE 3.1. PSNR COMPARISON

Test Sequences	Baseball	Claire_C	Coast Guard	Container	Flower Garden	Football	Foreman	Hall monitor	Mobile	Mother daughter	News	Salesman	Silent	Table Tennis
Line Doubling	20.66	33.89	24.78	24.45	18.48	24.68	28.19	25.82	18.33	30.87	26.45	28.13	28.30	24.41
Line Ave.	23.30	39.07	28.60	28.02	20.35	27.60	31.39	29.69	20.34	33.73	29.67	31.88	32.17	26.43
weave	21.68	44.46	28.22	**41.76**	18.99	20.93	29.86	33.97	**24.43**	39.60	38.70	39.80	35.63	25.01
ELA	22.59	38.50	28.08	27.15	20.10	27.13	32.14	28.92	20.22	33.08	27.41	30.61	31.40	26.66
Enhanced ELA	23.00	39.40	28.48	27.61	20.34	**27.58**	32.05	29.26	20.48	33.94	29.18	31.96	32.08	26.95
GMC	22.54	41.24	27.85	28.48	20.34	24.94	31.33	29.49	20.10	34.33	31.40	32.78	33.18	26.19
Proposed MC	**26.76**	**45.12**	**32.69**	41.14	**26.14**	27.23	**33.22**	**39.68**	24.12	**39.68**	**41.14**	**41.14**	**42.11**	**27.16**

According to subjective evaluation, the proposed method provides the best visual quality for all test sequences. Even the best intra-field methods produce flickering artifacts in areas with high spatial frequency; the proposed method has the least flickering artifacts in these areas, a consequence of the improvement of vertical resolution.

Figure 3.5 and Figure 3.6 show a picture from the Mobile and Flower Garden sequences deinterlaced by the proposed and the conventional methods. As it is shown in these figures, pictures deinterlaced by line averaging and Enhanced ELA methods are blurred, blocky and pixilated. In areas with high spatial frequency in both sequences, there is no vertical resolution improvement. The pictures deinterlaced by traditional motion compensation show some artifacts and the edges are not clear and smooth. The proposed method shows clearer and smoother edges than the other methods and improvement of vertical resolution in the pictures is obvious.

A flickering artifact is a common by product of intra-field deinterlacing methods. This artifact occurs when a sequence of images includes regions with high vertical spatial

frequency. In any one frame, aliasing in the high spatial frequency regions results in imperfect reconstruction. The flicker is produced by the different information from the odd and even frames. This effect can only be observed when a sequence of images is playing and thus cannot be shown adequately here in static form. However, subjective evaluation by observers has clearly demonstrated that the proposed method significantly reduces this flickering effect when compared to existing methods.

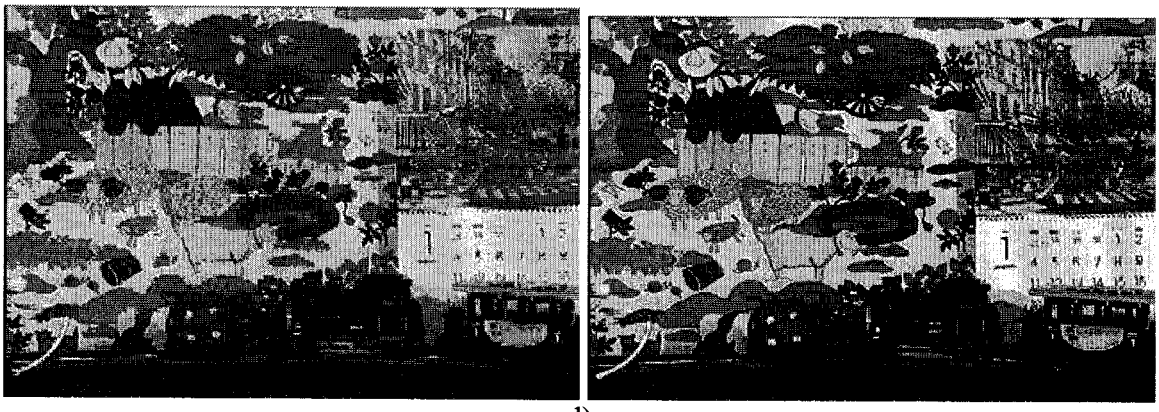
The proposed method resolves the covering and uncovering problem of the traditional motion compensated method and it removes the artifacts of the same parity motion estimators.

The results for different sequences deinterlaced by the proposed method and other methods can be found at [28].

3.5. Conclusion

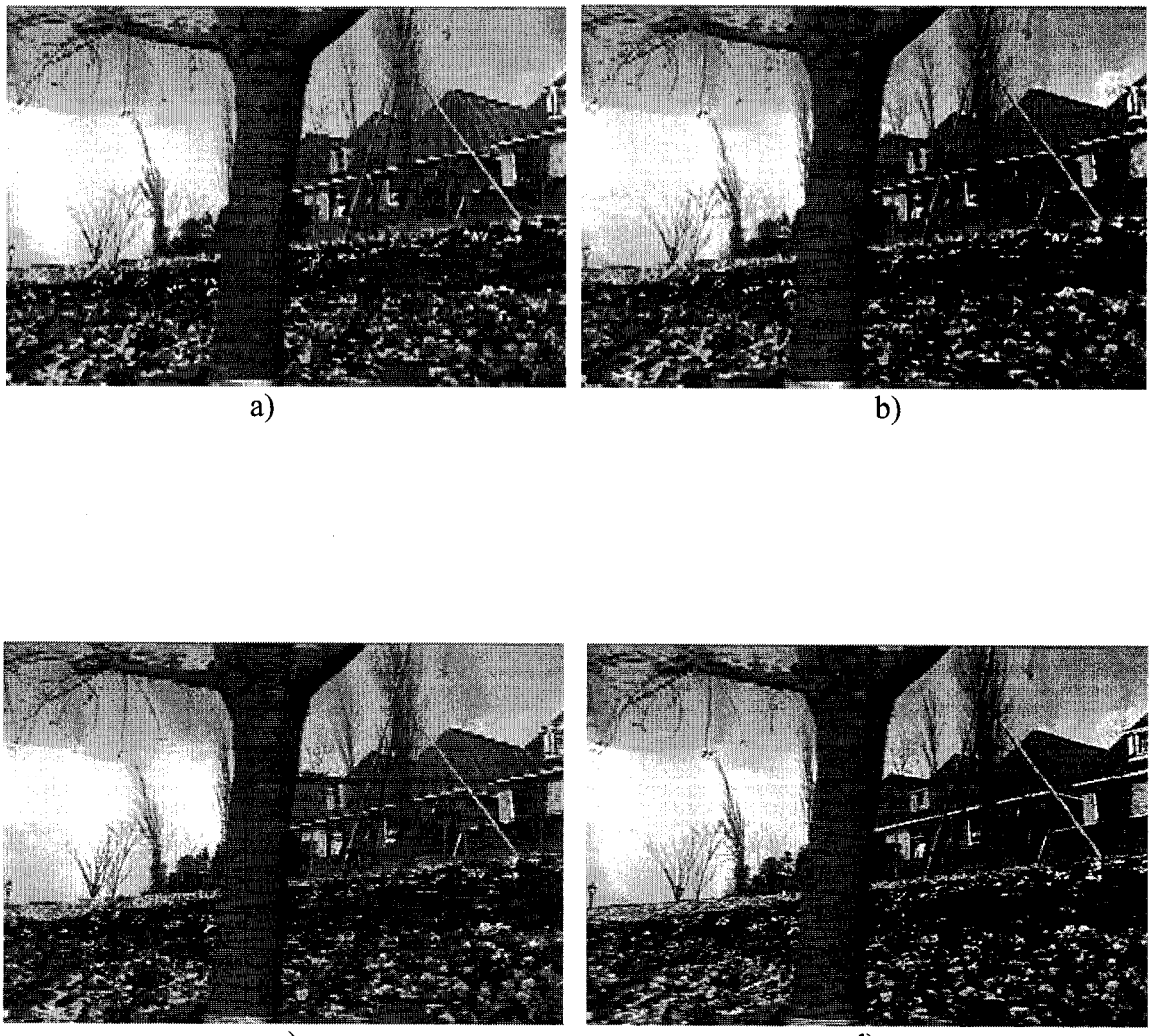
A new motion compensated method was proposed. The motion estimator uses the two previous de-interlaced frames and the next two pre-filtered frames to find one OMB and to calculate the motion vectors. The motion compensation method works based on the amount of vertical motion. It tries to calculate the missing lines from the existing lines to improve the vertical resolution and reduce flickering artifacts. Subjective and metric-based comparisons show that the proposed method compares favorably to previous approaches.

a) b)



d)

Figure 3.5. Subjective comparison of Mobile a) Line Averaging b) Enhanced ELA c) Traditional MC d)Proposed Method



a) b)

c) d)

Figure 3.6. Subjective comparison of Flower_Garden a) Line Averaging b) Enhanced ELA c) Traditional MC d) Proposed Method

3.6. References

[1] G. De Haan and E. B. Bellers, "Deinterlacing – An Overview," *Proc.of IEEE*, vol. 86, no. 9, pp.1839 – 1857, Sep. 1998.

[2] S. Pigeon and P. Guillotel, "Advantages and drawbacks of Interlaced and Progressive scanning formats," *CEC RACE/HAMLET Deliverable no R2110/WP2/DS/R/004/b1*, June 1995.

[3] Sung-Gyu Lee and Dong-Ho Lee, "A motion-adaptive de-interlacing method using an efficient spatial and temporal interpolation," *IEEE Trans. on Consumer Electronics,* vol. 49, no. 4, pp. 1266-1271, Nov. 2003.

[4] Y.Y. Jung, B.T. Choi, Yung-Jun Park and Sung-Jea Ko, "An effective de-interlacing technique using motion compensated interpolation," *IEEE Trans. on Consumer Electronics*, vol. 46, no. 3, pp. 460-466, Aug. 2000.

[5] K. Sugiyama and H. Nakamura, "A method of de-interlacing with motion compensated interpolation," *IEEE Transactions Consumer Electronics*, vol. 45, no. 3, pp. 611–616, August 1999.

[6] You-Young Jung, Byung-Tae Choi, Yung-Jun iark and Sung-Jea KO, "An Effective De-interlacing Technique Using Motion Compensated Interpolation," *IEEE Transactions on Consumer Electronics*, Vol. 46, No. 3, Aug. 2000, pp. 460-466

[7] A. Puri, H. M. Hang, and D. L. Schilling, "An efficient block matching algorithm for motion compensated coding," *ICASSP'87*, pp 1063-1066, Dalas, TX, 1987.

[8] S. Zhu and K. K. Ma, "A new diamond search algorithm for fast block matching motion estimation," *Proc. of Int. Conf. Information, Communications and Signal Processing,* vol. 1, pp. 292-6, 1997.

[9] R. Li, B. Zeng, and M. Liou, "A New Three-Step Search Algorithm for Block Motion Estimation," *IEEE Transactions Circuit and Systems for Video Technology*, vol. 4, NO. 4, pp. 438-442, Aug. 1994

[10] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Transactions Communication*, vol. COM-29, pp. 1799-1808, Dec. 1981

[11] M. Ghanbari, "The Cross-Search Algorithm for Motion Estimation," *IEEE Transactions Communication*, vol. 38, NO. 7, pp. 950-953, July 1990

[12] L.K. Liu and E. Feig, "A Block-Based Gradient Descent Search Algorithm for Block Motion Estimation in Video Coding," *IEEE Transactions Circuit and Systems for Video Technology*, vol. 6, NO. 4, pp. 419-422, Aug. 1996

[13] S. Zhu and K.K. Ma, "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation," *IEEE Transactions Image Processing,* vol. 9, NO. 2, pp. 287-290, Feb. 2000

[14] Jiancong Luo, Ishfaq Ahmad1 and Xizhang Luo, "An Adaptive Cross Search Algorithm for Block Matching Motion Estimation," *IEEE ICCCAS*, June 2004, pp.914-918

[15] R. Srinivasan and K. R. and K. R. Rao, "Predictive coding based on efficient motion estimation, " *IEEE Transactions Communication*, vol. COM-33, pp. 1011-1014, Sept. 1985

[16] Xuan-Quang Banh and Yap-Peng Tan "Efficient Video Motion Estimation Using Dual-cross Search Algorithms" , *IEEE 2005*, pp. 5485-5488.

[17] H. Mahvash M., J.M. P. Langlois, and Y. Savaria, "A Threshold-Based De-Interlacing Algorithm Using Motion Compensation and Directional Interpolation," *IEEE ICECS*, Nice France, Dec. 2006.

[18] Seungclian Byun, Jeongmoon Byun, and Gyeonghwan Kim, "A Recursive Approach For De-interlaacing using Improved ELA and Motion Compensation Based Bi-directional BMA," *International Conference on Image Processing (ICIP)* 2004, pp. 1679-1682

[19] Xinbo Gao, Juxia Gu, Jie Li, "De-interlacing Algorithms Based on Motion Compensation," *IEEE Transactions on Consumer Electronics*, Vol. 51, No. 2, MAY 2005, pp. 589-599

[20] Rui-Jie Xiao, AI&R, Xi'an Jiaotong, "De-interlacing with motion compensation and edge-dependent interpolation in complement using judder pattern," *ICCE 2005*, pp. 81-82 (1-d)

[21] D. J. Wang and J. J. Leou, "A new approach to video format conversion using bidirectional motion estimation and hybrid error concealment:' *Journal of information Science and Engineering,* vol. 17, no. *5,* pp. 763-777, 2001.

[22] Jelena Kova˘cevi´c, Robert J. Safranek, and Edmund M. Yeh, "Deinterlacing by Successive Approximation," *IEEE Transactions on Image Processing*, vol. 6, no. 2 Feb. 1997, pp. 339-344,

[23] Min Kyu Park and Moon Gi Kang, "New Global Motion Compensated De-interlacing Algorithm Based on Horizontal and Vertical Patterns," *ICASSP 2004*, pp. III345-

[24] Yu-Lin Chang, Shyh-Feng Lin, Ching-Yeh Chen, and Liang-Gee Chen, "Video De-Interlacing by Adaptive 4-Field Global/Local Motion Compensated Approach," *IEEE Transactions On Circuits and Systems for Video Technology*, vol. 15, no. 12, Dec. 2005, pp. 1569-1582

[25] Ohjae Kwon, Kwanghoon Sohn, and Chulhee Lee, "Deinterlacing using Directional Interpolation and Motion Compensation," *IEEE Transactions on Consumer Electronics*, Vol. 49, No. 1, Feb. 2003, pp. 198-203

[26] Motion Compensation Assisted Motion Adaptive Interlaced-to-Progressive Conversion, Seungjoon Yang, You-Young Jung, Young Ho Lee, and Rae-Hong Park, , *IEEE Transactions On Circuits and Systems for Video Technology*, vol. 14, no. 9, Sep. 2004, pp. 1138-1148

[27] "Reliable Motion Detection/Compensation for Interlaced Sequences and Its Applications to Deinterlacing, Renxiang Li, Bing Zeng, and Ming L. Liou, *IEEE Transactions On Circuits and Systems for Video Technology*, vol. 10, no. 1, Feb. 2000 , pp. 23-29

[28] http://www.grm.polymtl.ca/~savaria/Deinterlaced_sequences/

Chapter 4

A Pattern-Based Directional Interpolation Deinterlacing Algorithm

Hossein Mahvash Mohammadi, J.M. Pierre Langlois, *Member*, IEEE, and Yvon Savaria, *Fellow*, IEEE

Abstract *— A Pattern-Based Directional Interpolation (PBDI) deinterlacing algorithm is proposed in this paper. The algorithm compares three-pixel patterns in the upper and lower lines with respect to the target pixel, to detect edge directions. Directional interpolation with median filtering is used in the direction with the highest correlation to calculate missing pixels. Three patterns in each line are compared to seven patterns in the other line. This allows matching twenty-one possible different combinations of edge directions and positions, while providing improved directional interpolation. This also improves the discrimination between actual and misleading edges. Pattern comparison is done by a combination of the sum of absolute differences and a gradient-based difference. This improves pattern comparison accuracy in situations where the sum of absolute differences alone does not work well. Experimental results show that the PBDI performs better than traditional edge-based methods, based on objective and subjective criteria.*

Index Terms — Deinterlacing, Edge-based methods, Directional Interpolation, Pattern comparison.

4.1. Introduction

Video interlacing, where only odd or even lines from a frame are transmitted, was originally proposed to decrease the transmission bandwidth by a factor of two [1]. Interlacing is a down-sampling procedure in the spatial domain, performed without pre-filtering, and for which the Nyquist criterion is in general not satisfied. Consequently, reconstructing a perfect image may not be possible in some cases. Nevertheless, interlacing is still part of many video standards, including HDTV. Deinterlacing is necessary in order to display interlaced video on a progressive scan display.

Video sequences tend to be highly correlated in the spatial and temporal domains. Deinterlacing methods exploit this fact to estimate missing pixels, restoring vertical resolution, while trying to avoid the introduction of artifacts. They can be classified into intra-field, inter-field, motion adaptive and motion compensated methods.

Intra-field deinterlacing methods use spatial interpolation. Missing pixels are calculated based on pixels in their neighborhood. Intra-field methods are the least computationally expensive, but they can have poor performance in areas with high spatial frequency where they tend to produce flickering artifacts. Line repetition and line averaging are the simplest intra-field methods.

Inter-field deinterlacing methods use temporal interpolation. They can perfectly restore vertical resolution in still areas. In moving areas, however, they are susceptible to producing annoying artifacts such as combing. Motion adaptive deinterlacing methods combine the advantages of inter-field and intra-field methods in the still and motion areas, respectively. A motion detection algorithm is necessary to differentiate between

still and motion areas.

Motion compensated methods are the most powerful and complex deinterlacing methods. They exploit the advantages of inter-field methods in the motion and static areas. A motion estimator calculates the spatial displacement for blocks in the current field, and temporal interpolation is performed in the direction of the motion vectors.

All advanced deinterlacing methods frequently perform intra-field deinterlacing. For example, this happens in a hybrid motion compensation deinterlacing system, when no matching block can be found for a given block. Consequently, there is always a requirement for improved intra-field deinterlacing methods.

In this paper, we introduce a new intra-field deinterlacing algorithm that we call Pattern-Based Directional Interpolation (PBDI). The paper is organized as follows. In section 4.2, an overview of edge-based deinterlacing methods is presented and their shortcomings are summarized. Section 4.3 presents the proposed PBDI deinterlacing method. Section 4.4 shows experimental results and a comparison with previous methods, and section 0 concludes the paper.

4.2. Intra-Field Edge-Based Deinterlacing

4.2.1. Basic edge-based deinterlacing methods

Edge-based deinterlacing methods use neighboring pixel values to assess the edge direction in the missing pixel position and they interpolate toward that direction. They often avoid the 'jagged edge' effect in the presence of straight edges in the image, but their performance can be affected by complex patterns and textures. Their computational

complexity and buffering requirements are modest, and they normally achieve the best quality in the intra-field deinterlacing category.

Several edge-based deinterlacing methods have been proposed. Edge-based Line Average (ELA), illustrated in Figure 4.1, is the simplest edge-based deinterlacing method [2] and one of the most popular because it involves simple calculations and provides satisfactory results. It can perform interpolation along three directions only, and typically fails to provide good quality in the presence of near-horizontal edges.

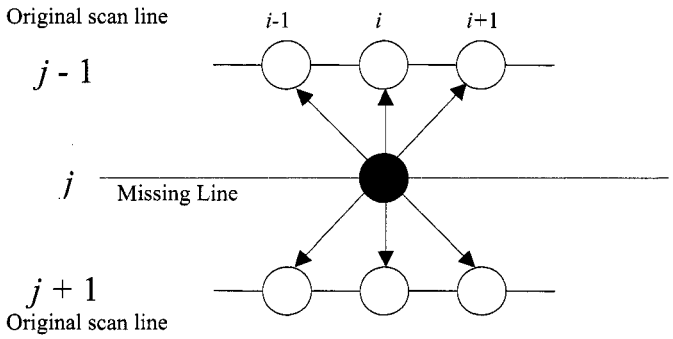


Figure 4.1. Pixels and directions used in ELA

Three directions are considered by calculating the corresponding absolute differences between pairs of pixels. The direction with minimum brightness difference is selected as the most probable edge:

$$k = \arg \underset{-1 \le k \le 1}{Min} \left| f(i-k, j-1) - f(i+k, j+1) \right| \qquad (4.1)$$

where f is the intensity of a pixel, i and j are the pixel's column and row, respectively, and k is the direction with the highest directional correlation. The missing pixel is calculated by directional interpolation toward direction k as follows:

$$f(i, j) = 1/2\left[f(i-k, j-1) + f(i+k, j+1) \right] \qquad (4.2)$$

Improved versions of this method have been proposed. In Modified ELA [3], edge direction detection is performed based on five directions instead of three, as shown in Figure 4.2. Edge detection is decomposed in two steps. In the first step, the absolute difference between pixels at coordinates $(i, j-1)$ and $(i, j+1)$ is calculated. If the difference is larger than a given threshold, then it is assumed that the missing pixel is inside an edge. This check may fail in some situations, such as for one-pixel wide edges and for edges that contain complex patterns.

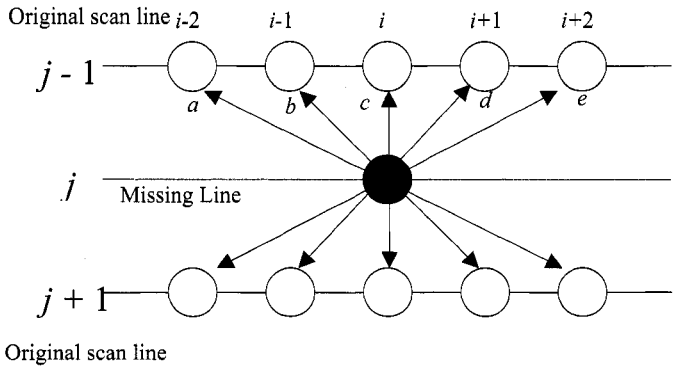


Figure 4.2. Pixels and directions used in Modified and Enhanced ELA

If an edge is present, then the second step consists of checking whether the edge is dominant. The absolute difference of the pair of pixels that are on either side of the edge is calculated. If the absolute difference of these two differences (along the edge and on either side of the edge) is greater than a second threshold, then the edge is considered dominant. In this case, directional interpolation is performed. Otherwise, vertical interpolation is used.

Modified ELA may fail in the presence of narrow and near-horizontal edges. It is also susceptible to misleading edges due to high correlation in background pixels.

Enhanced ELA [4] is identical to Modified ELA, except in the last step. The missing pixel is calculated as the median of three values: the value of the directional interpolation, the upper pixel and the lower pixel. Edge-Dependent Deinterlacing (EDDI) [5] is another method that performs directional interpolation only for dominant edges of an image.

4.2.2. Misleading edges

In edge-based methods, misleading edges occur mostly in the presence of narrow edges over highly correlated backgrounds. In such a situation, the algorithm may incorrectly interpolate the missing pixel based on background data instead of edge data. Figure 4.3 shows this situation for ELA, when the edge direction *a* can be confused with *b*. If the absolute brightness difference between $f(i\text{-}1, j\text{-}1)$ and $f(i+1, j+1)$ is smaller than that of $f(i+1,j\text{-}1)$ and $f(i\text{-}1,j+1)$, then the directional interpolation will be done (incorrectly) in direction *b* instead of *a*. For example, this can occur if the edge is textured or if its intensity varies, possibly due to a lighting gradient, and if the background is homogeneous.

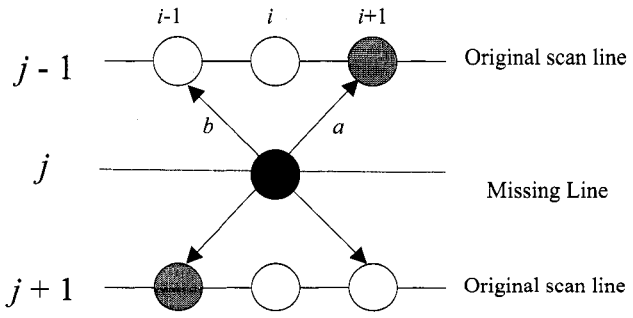


Figure 4.3. Misleading edge in ELA

The Modified and Enhanced ELA algorithms can also suffer from misleading edges.

Dominant edge recognition can fail for narrow edges, even if the background data are not more correlated than the edge data, because it relies on high correlation for pixels in the edge direction and low correlation for pixels in the opposite direction. This assumption tends to be true for wide edges but false for narrow edges. Figure 4.4 shows two examples of this situation, with edge widths of three and five pixels. In both cases, highly correlated background data in the opposite direction of the edges (shown by direction *b*) make them be classified as non-dominant. In such cases, the Modified and Enhanced ELA algorithms perform vertical interpolation even though definite edges are clearly present.

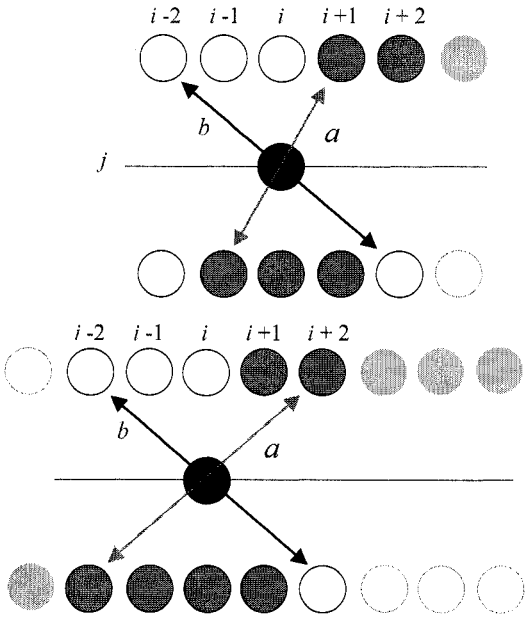


Figure 4.4. Correlated background data in a direction opposite to that of the true edge in Modified and Enhanced ELA can cause misleading edges

4.2.3. Improving edge angle resolution

Using few neighboring pixels, e.g. 6 in the case of ELA and 10 in the case of Modified ELA, Enhanced ELA and EDDI may cause misleading or inaccurate edge

direction estimation. In order to improve edge angle resolution, several authors have proposed extending the search horizontally and/or vertically.

The Directional-Oriented Interpolation (DOI) method [6] performs a pattern-based comparison to detect the edge direction. As shown in Figure 4.5, it uses two upper and two lower reference lines to detect edges. A 2 × 3 pixels pattern is centered on the missing pixel. A search is performed for the most similar patterns in distinct search windows above and below the missing pixel. In each search window, the most similar patterns produce an upper and a lower spatial direction vector (SDV). If the two SDVs are aligned, directional interpolation is performed toward the underlying edge. Vertical averaging is performed when the SDVs are not aligned and when the absolute difference between the upper and lower pixel is less than a given threshold. DOI increases edge direction resolution and performs well in images with strong edges. In [6], the authors present results for a case where the horizontal search range was 9 pixels wide on either side of the missing pixel.

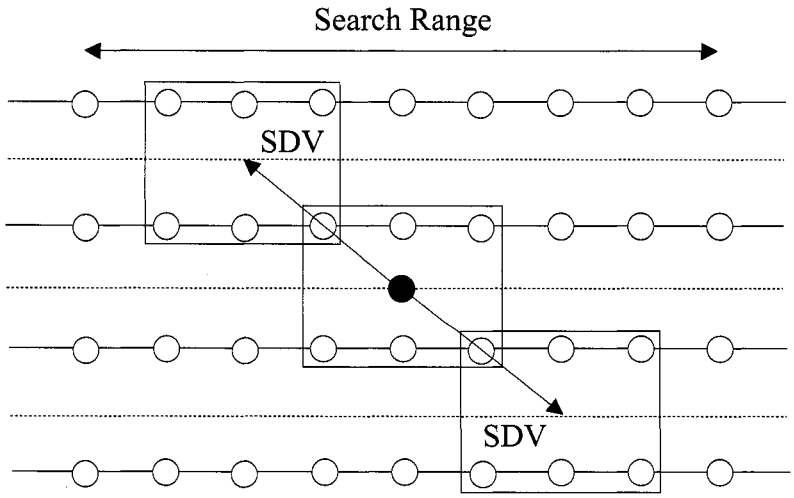


Figure 4.5. The DOI method

A generalized model to dynamically extend the number of edge direction was proposed by Chang et al., with tradeoffs made between angular resolution and computational complexity [7]. The same authors also proposed an edge-based method that uses two upper and lower scan lines to detect seventeen edge directions [8].

The nature of the deinterlacing problem dictates that only rows 1, 3, 5, etc., above or below the missing pixel can be used. While appealing from an edge direction perspective, using additional rows implies a reliance on data that is progressively further away from the missing pixel, and consequently that can be much less correlated.

In areas with low spatial frequency and with homogeneous textures, using a large number of neighboring pixels can improve edge direction detection. However, in areas with high spatial frequency, using pixels that are widely separated can lead to incorrect edge direction estimation. The design therefore faces a tradeoff between edge angular resolution and information correlation.

4.2.4. Spatio-temporal methods

Some edge-based methods use data from neighboring fields for edge direction detection and temporal interpolation. These methods combine spatial and temporal information. Recent examples include the Spatio-Temporal ELA (STELA) [9], Fuzzy Weighted Filtering (FWF) [10] and the inpainting-based method [11]. Compared to intra-field edge-based methods, the spatio-temporal methods tend to be significantly more complex, especially in terms of buffering requirements.

4.3. Proposed PBDI Algorithm

4.3.1. Overview of the proposed PBDI algorithm

Most edge-based deinterlacing methods compare single pixels in the lower and upper lines to estimate the edge direction. In areas with high spatial frequency and with complex patterns, this can cause incorrect edge direction detection. To decrease the probability of this occurrence, the proposed PBDI algorithm uses pattern comparison instead of single pixel comparison. Most of the paper assumes a pattern size of 3 pixels, even though wider patterns can be considered and some of the key equations and relations describing a generalized PBDI will be stated for a pattern size of n, an odd integer larger or equal to 3. A pattern in position *(i, j)* is defined by a vector of three horizontally contiguous pixels as follows:

$$P(i,j) = [f(i-1,j),\ f(i,j),\ f(i+1,j)] \qquad (4.3)$$

where P refers to the pattern, f is the input field pixel intensity, and i and j point to the horizontal and vertical positions, respectively.

Two search windows are defined, one in each of the lines immediately above and below the missing pixel. The search windows are nine pixels wide and are centered horizontally on the missing pixel. Figure 4.6 illustrates the situation. The missing pixel is at position (i, j). Two patterns are shown defining an edge angle, one in each of the upper and lower search windows.

There are seven possible patterns in each search window. $P(i\text{-}3, j\text{-}1)$ to $P(i\text{+}3, j\text{-}1)$ are patterns in the upper search window and $P(i\text{-}3, j\text{+}1)$ to $P(i\text{+}3, j\text{+}1)$ are in the lower

search window. Taking one pattern from each search window, pairs of patterns are compared. The two most similar patterns are called matching patterns, and they define an edge direction. In Figure 4.6, edge a crosses the center pixel of two matching patterns.

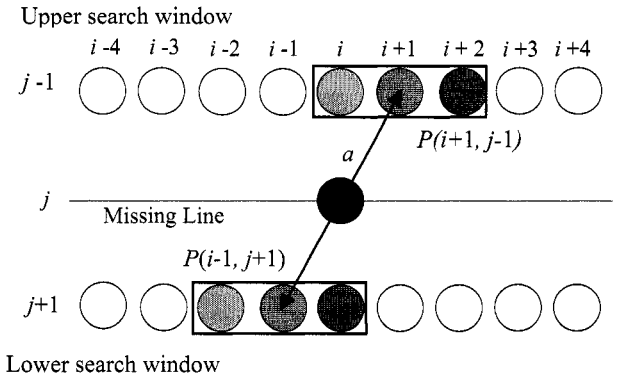


Figure 4.6. Pattern comparison in the PBDI

4.3.2. Increased number of edge directions and angle resolution

The PBDI algorithm greatly increases the possible number of detected edges to the twenty-one combinations of directions and positions shown in Figure 4.7. Each considered edge is assumed to go through the center pixel of the upper and lower patterns.

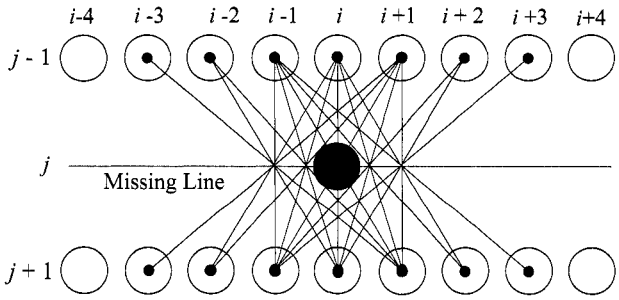


Figure 4.7. Edge positions and directions detected by the PBDI

Furthermore, the PBDI algorithm is able to detect edges that do not cross missing

line j exactly at the missing pixel position. Figure 4.8 shows an example where the edge crosses line j at position $i - 0.5$. Edge based methods that compare single pixels may unable to achieve this resolution.

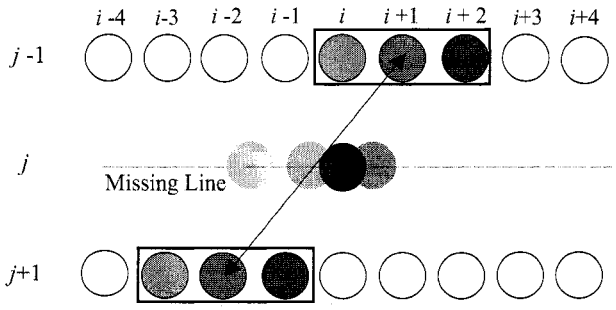


Figure 4.8. Edge crossing within missing pixels

An inspection of Figure 4.7 reveals that not all possible pattern pairs are considered to find a matching pattern. For 3-pixel patterns and 9-pixel windows, there are seven patterns in each upper and lower lines and forty-nine different pattern pairs. Twenty pattern pairs define edges that do not intersect the missing pixel at all and from which no useful data could be used for interpolation. Eight other pairs are also excluded from the comparisons to decrease the occurrence of misleading edges, as further explained in section 4.3.3. This leaves the twenty-one useful pattern pairs shown in Figure 4.7.

4.3.3. Resistance to misleading edges in background data

The proposed PBDI algorithm is resistant to misleading edges arising from situations where background data are more correlated than edge data. This increased resistance is achieved by including the pixel immediately above or the pixel immediately

below the missing pixel, or both, in all comparisons. To satisfy this constraint, only three patterns in either search window are compared with all other patterns in the other search window. This resolves the misleading edge problem for all edges crossing the pixels immediately above or below the missing pixel. However, a misleading edge can still happen if the true edge does not cross the pixels immediately above or below the missing pixel and if background data is more correlated than edge data.

Figure 4.9 shows this situation with patterns $P(i\text{-}1, j\text{-}1)$, $P(i, j\text{-}1)$ and $P(i\text{+}1, j\text{-}1)$ in a limited upper search window being compared with all patterns in the lower search window. The patterns $P(i\text{-}3, j\text{-}1)$, $P(i\text{-}2, j\text{-}1)$, $P(i\text{+}2, j\text{-}1)$ and $P(i\text{+}3, j\text{-}1)$ are only compared with the three patterns in the limited lower search window, shown in Figure 4.10. The same limitation is also applied for the patterns in the lower search window when they are compared with all patterns in the upper search window.

Figure 4.9 shows an edge crossing the pixel above the missing pixel. The presence of pixel $(i, j\text{-}1)$ in all compared patterns in the upper search window has significant impact on the similarity calculation. This generally prevents homogeneous background data from having a higher similarity than the real edge. ELA, Modified ELA and Enhanced ELA would fail to recognize the edge direction and would find a misleading edge in this situation.

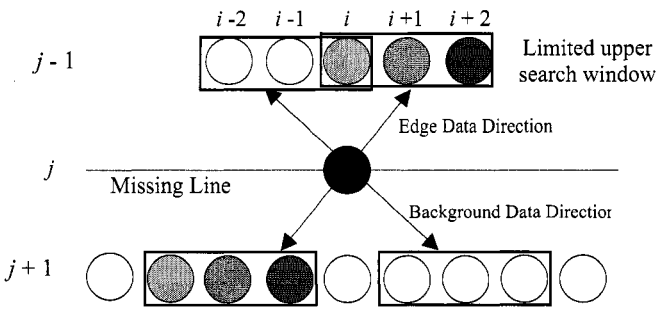


Figure 4.9. Edge and background data direction detection in edges with three pixels width

Figure 4.10 also shows the presence of pixel $(i, j+1)$ in all compared patterns in the lower search window that causes a large error when it is compared with pattern $P(i+2, j-1)$. This is in spite of the fact that the background pixels all have the same intensity.

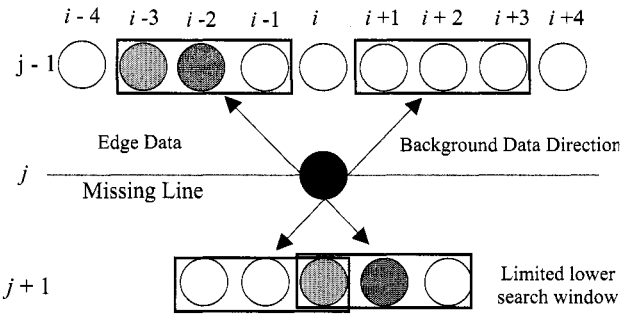


Figure 4.10. Edge and background data direction detection in edges with two pixels width

In order to exploit this feature, the limited upper and lower search windows' width must always be twice the pattern size minus one. Consequently, all patterns in the limited windows always include the pixels that are immediately above or below the missing pixel.

4.3.4. Criterion for matching patterns selection

In video coding, different criteria such as the Sum of Absolute Differences (SAD) and phase correlation have been proposed to find the best matching blocks [13]. The SAD is calculated pixel-by-pixel from a reference block and blocks in a search image window. It is fairly simple to calculate, and only involves sums and differences. Phase correlation is based on the Fourier transform of blocks and can provide better results than the SAD. However, it involves much greater computational complexity.

In the PBDI algorithm, a combination of SAD and gradient-based difference (GBD) is used for pattern comparison. GBD consists of calculating the intensity differences between the center pixel and its neighbors on either side. To our knowledge, no other pattern based deinterlacing method exploits the GBD. These differences are then compared for each pair of patterns under consideration. In some situations, the GBD provides better results than the SAD. This includes the comparison of patterns with the same texture but different brightness levels.

Figure 4.11 shows such a situation. Typical patterns for three contiguous sequential pixels are shown in Figure 4.11.a to 4.11.d. In Figure 4.11.e, the brightness difference between two flat patterns is shown. Figure 4.11.f compares the brightness difference between two other patterns. According to the SAD, the patterns of Figure 4.11.f are more similar than the patterns of Figure 4.11.e. Intuitively, however, the patterns in Figure 4.11.e are more similar because they only differ in terms of brightness and not in terms of texture. The patterns in Figure 4.11.e would have zero GBD while the patterns of Figure 4.11.f that would have a relatively high GBD.

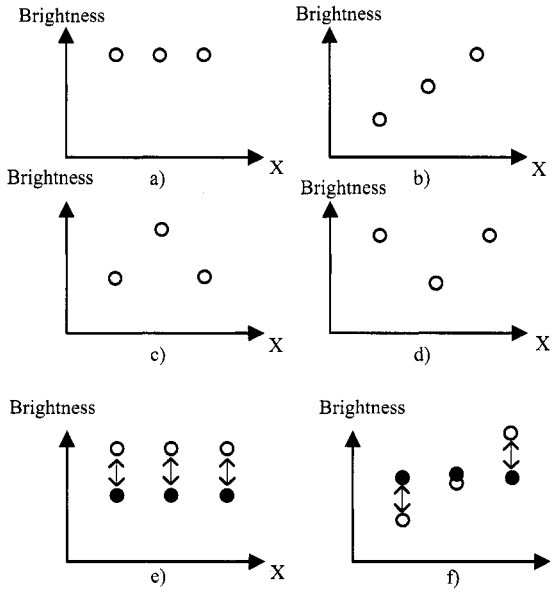


Figure 4.11. Patterns of three pixels and comparison

To clarify this idea, Figure 4.12 shows a pattern in the upper line and two candidate patterns in the lower line. The pattern on the right potentially corresponds to an edge whose intensity increases uniformly from 70 to 80 along direction *a*. However, using SAD alone, the pattern in direction *b* is a better match although it includes different pixel-to-pixel variations from the pattern in the upper line. Combining GBD with SAD can compensate for this effect. Based on these observations, we propose to select matching patterns based on a combination of SAD and GBD. The error between two patterns is calculated as follows:

$$Error(P_u, P_l) = \alpha .\ SAD(P_u, P_l) + (1-\alpha)\ .\ GBD(P_u, P_l) \qquad 0 < \alpha < 1 \tag{4.4}$$

where GBD is the gradient-based difference and α is a coefficient that determines the relative proportions of SAD and GBD. P_u and P_l are patterns in the upper and lower

search windows, respectively.

The SAD for patterns *P(u, j-1)* and *P(l, j+1)*, abbreviated by P_u and P_l, is calculated as follows:

$$SAD(P_u, P_l) = \sum_{k=-1}^{1} |f(u+k, j-1) - f(l+k, j+1)| \qquad (4.5)$$

For a pair of patterns being compared, we define dif_l (left difference) as the difference in brightness variations between the middle pixel of each pattern and their left neighbors. We further define dif_r (right difference) to be the equivalent on the right side. Equations 4.6 and 4.7 show the calculation details of dif_l and dif_r for patterns P_u and P_l at positions $(u, j\text{-}1)$ and $(l, j+1)$ respectively.

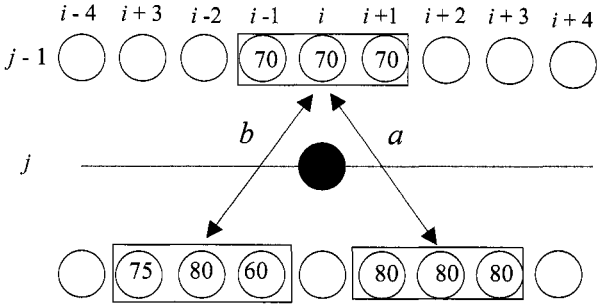


Figure 4.12. SAD and GBD comparison

$$dif_l(P_u, P_l) = [f(u-1, j-1) - f(u, j-1)] - [f(l-1, j+1) - f(l, j+1)] \qquad (4.6)$$

$$dif_r(P_u, P_l) = [f(u, j-1) - f(u+1, j-1)] - [f(l, j+1) - f(l+1, j-1)] \qquad (4.7)$$

The GBD for patterns P_u and P_l is calculated as follows:

$$GBD(P_u, P_l) = \left|dif_l(P_u, P_l)\right| + \left|dif_r(P_u, P_l)\right| \tag{4.8}$$

The GBD can also be defined for pattern size n, with $n>3$, as given by (4.9).

$$\begin{aligned} GBD(P_u, P_l) = \sum_{k=-n/2}^{n/2-1} \Big| & \left[f(u+k, j-1) - f(u+k+1, j-1)\right] \\ & - \left[f(l+k, j+1) - f(l+k+1, j+1)\right] \Big| \end{aligned} \tag{4.9}$$

The parameter α in (4.5) that sets the relative importance of GBD and SAD was determined empirically. The PBDI algorithm was applied to nine video test sequences for different values of α in the range [0, 1]. Figure 4.13 shows the normalized PSNR results. Smaller values of α increase the GBD contribution while larger ones increase the SAD contribution. According to our experiments, a value of α around 0.4 provides the best compromise. For ease of implementation we selected α equal to 0.5 that provides a performance close to the optimal one (0.04 dB degradation in the sum of PSNR over 9 video sequences). Indeed, in this case, the multiplications can be replaced by trivial shifts. It is of interest that the curves in Figure 4.13 are normalized according to their respective maximum values, therefore the best α based on the overall PSNR cannot be selected visually as Figure 4.13 misleadingly show the best α around 0.25.

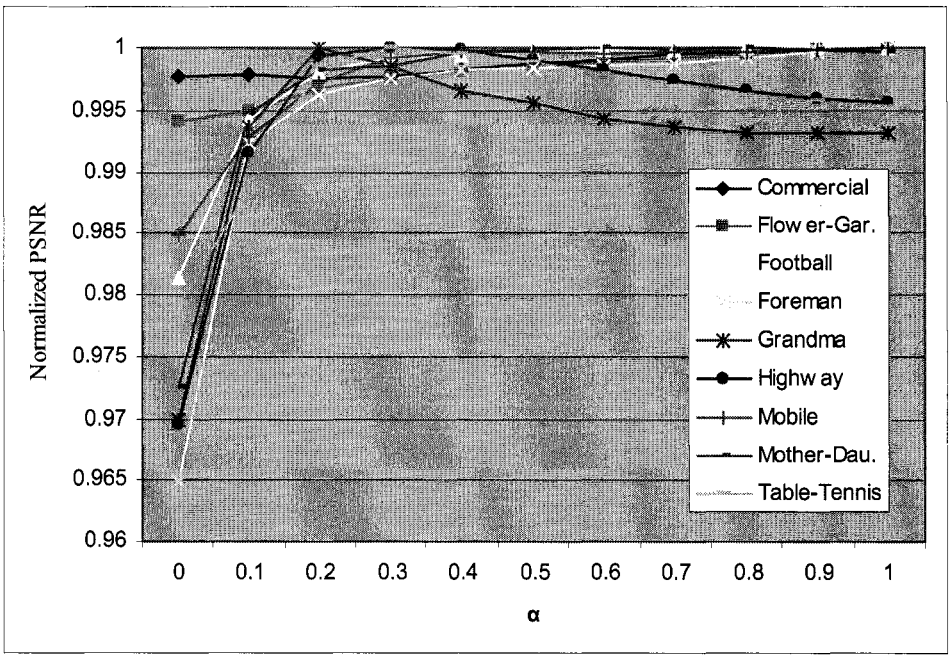


Figure 4.13. Normalized PSNR versus α in the PBDI algorithm

4.3.5. Missing pixel interpolation

The edge defined by matching patterns is specified by their horizontal displacement from the missing pixel. For the case considered so far, we have:

$$(u,l) = \arg \min_{\substack{|u+l|\le 2 \\ |u|\le 1 \text{ or } |l|\le 1}} \left(Error(P(i+u, j-1), P(i+l, j+1)) \right) \quad (4.10)$$

where u and l represent the upper and lower patterns horizontal displacement from the missing pixel position at (i, j), respectively. As explained previously, some combinations of u and l are not useful, because they result either in edges that do not go through the missing pixel, or involve patterns that contribute interpolation data subject to increased chances of selecting data from misleading edges for interpolation. The condition $|u + l| \le 2$ guarantees that the detected edge passes within a distance of one

pixel from the missing pixel. The conditions $|u| \leq 1$ and $|l| \leq 1$ are used to exclude pattern comparisons that may cause misleading edges, as explained in section 4.3.3.

For acceptable values of u and l, the directional interpolation value for the missing pixel is calculated by (4.11).

$$f(i,j) = \begin{cases} (f(i+u-l, j-1) + f(i-u+l, j+1))/2 & (u-l) \bmod 2 = 0 \\ (f(i+\lfloor u-l \rfloor, j-1) + f(i+\lfloor u-l \rfloor + 1, j-1) & \\ \quad + f(i-\lfloor u-l \rfloor, j+1) + f(i-\lfloor u-l \rfloor - 1, j+1))/4 & \text{otherwise} \end{cases} \tag{4.11}$$

This version of the equation is for the specific case of three-pixel patterns. The various possibilities can be classified into three broad classes depending on the alignment of the edge with the missing pixel:

a. the edge goes through the missing pixel;

b. the edge goes through either the left or right neighbor of the missing pixel; or,

c. the edge crosses the missing pixel line exactly between the missing pixel and one of its immediate neighbors.

In the first case, the edge goes through the missing pixel. Its intensity is calculated as the average of the two center pixels of the matching patterns. These two pixels are aligned with the missing pixel in the direction of the edge. In the second case, the edge goes through either the missing pixel's left or right immediate neighbors. The missing pixel intensity is then calculated as the average of the right or left pixels, respectively, in the matching patterns. In the third case, the edge crosses the missing pixel line between

the missing pixel and its left or right immediate neighbors. A straight line can be drawn through the missing pixel with a direction parallel to the edge. This line crosses the matching patterns between two pixels: the center pixel and either the right or left pixels, respectively. The missing pixel's intensity is then calculated as the average of these four pixels. The final missing pixel intensity is taken as the median of $f(i, j - 1), f(i, j + 1)$ and the result of the directional interpolation.

Using a geometric argument, it can be shown that (4.11) results in full bilinear interpolation since the interpolated pixel always lines up with the data selected for directional interpolation or falls exactly in the middle of the lines linking the corner the interpolation parallelogram.

4.4. Analysis of the Results

4.4.1. Objective and Subjective Evaluations

Ten progressive sequences of 30 frames each were used to evaluate the performance obtained with a software implementation of the proposed PBDI algorithm. The sequences were first interlaced and then deinterlaced with different methods. In order to obtain objective performance comparisons, the Mean Squared Error (MSE) and the Power Signal to Noise Ratio (PSNR) were measured in each case.

The MSE and PSNR equations are expressed as follows:

$$MSE = \frac{1}{X.Y} \sum_{i=1}^{X} \sum_{j=1}^{Y} (f(i, j) - \hat{f}(i, j))^2 \qquad (4.12)$$

$$PSNR = 10.\log(\frac{255^2}{MSE}) \qquad (4.13)$$

where f and $\hat{f}$ refer to the original and deinterlaced images, respectively, and X and Y are the image dimensions.

Table 4.1 lists the results. For the PBDI algorithm, the value of α was set to 0.5. For DOI, its search range was set from -4 to +4. The PBDI method achieved an average improvement of 0.19 dB over DOI, an average of 0.08 dB over Enhanced ELA, an average of 0.47 dB over ELA, an average of 0.28 dB over the line averaging, and an average of 2.47 dB over the line repetition method.

TABLE 4.1. PSNR COMPARISON BETWEEN DIFFERENT METHODS (IN DB)

Test Sequences	Line Rep.	Line Ave.	ELA	Enh. ELA	DOI	PBDI
Commercial	18.62	20.51	20.07	20.41	20.39	**20.52**
Flower	18.45	**20.25**	19.95	20.19	20.24	20.13
Football	24.95	27.78	27.35	27.80	27.73	**27.81**
Foreman	24.74	27.36	26.81	27.52	**27.70**	27.51
Grandma	30.61	34.63	34.47	**35.12**	34.66	34.96
Highway	29.36	32.00	32.76	32.63	32.60	**32.91**
Mobile	18.16	20.16	20.00	20.28	20.15	**20.47**
Mother Daughter	31.01	33.52	33.04	**33.77**	33.67	33.67
Table Tennis	22.23	24.05	24.18	24.49	23.96	**24.85**
Overall	24.24	26.70	26.51	26.90	26.79	**26.98**

Some experiments were conducted regarding the best pattern size for PBDI. However, the pattern size of three pixels used in the proposed PBDI algorithm achieved

the best results according to MSE. The best pattern size may differ for different sequences. We conjecture, however, that in spite of the expected advantage of considering more angles and edge positions, a wider pattern size may result in more incorrect edge detections, because of reduced correlation as the distance increases between the missing pixel and the considered patterns.

It is well known that the popular PSNR metric, widely used because of its ease of use, can be a very misleading indicator of the performance of a video processing algorithm. Subjective evaluations were thus conducted. Some results of those subjective evaluations are shown in Figures 4.14 and 4.15. They show portions of frames of the "Football" and "Foreman" sequences deinterlaced by different methods. As shown in these pictures, the PBDI algorithm produces visually clearer and smoother edges, and in general images of noticeably better quality. This is especially true around the player's thigh in Figure 4.14 and around the hat in 4.15. More importantly, for video sequences, the PBDI algorithm produces fewer flickering artifacts than the other edge-based methods. Results for various sequences deinterlaced by the PBDI algorithm and other methods are available on-line [15].

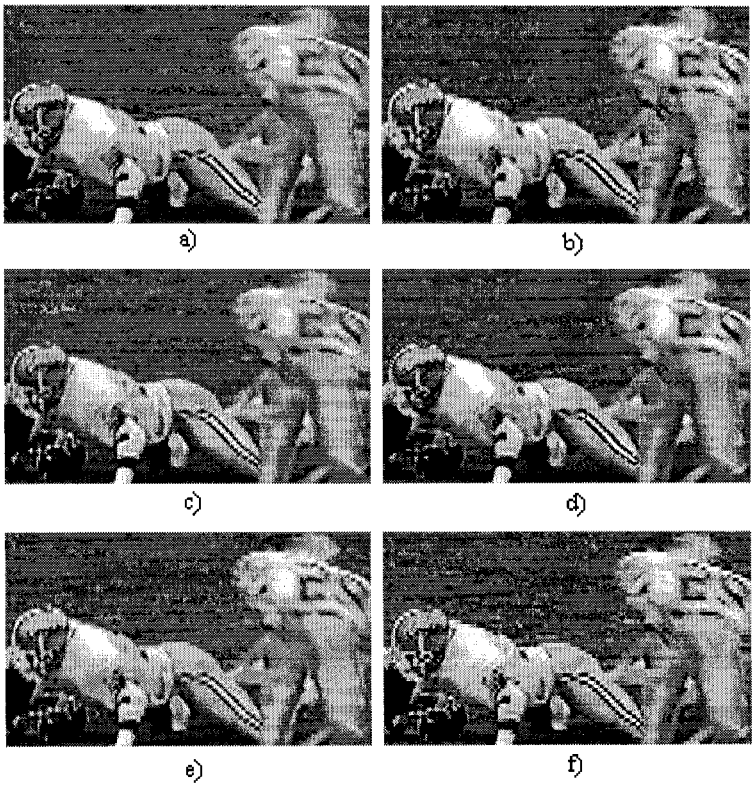


Figure 4.14. Football picture deinterlaced by a) PBDI b) DOI
c) Enhanced ELA d) ELA e) Line Averaging f) Line Repetition

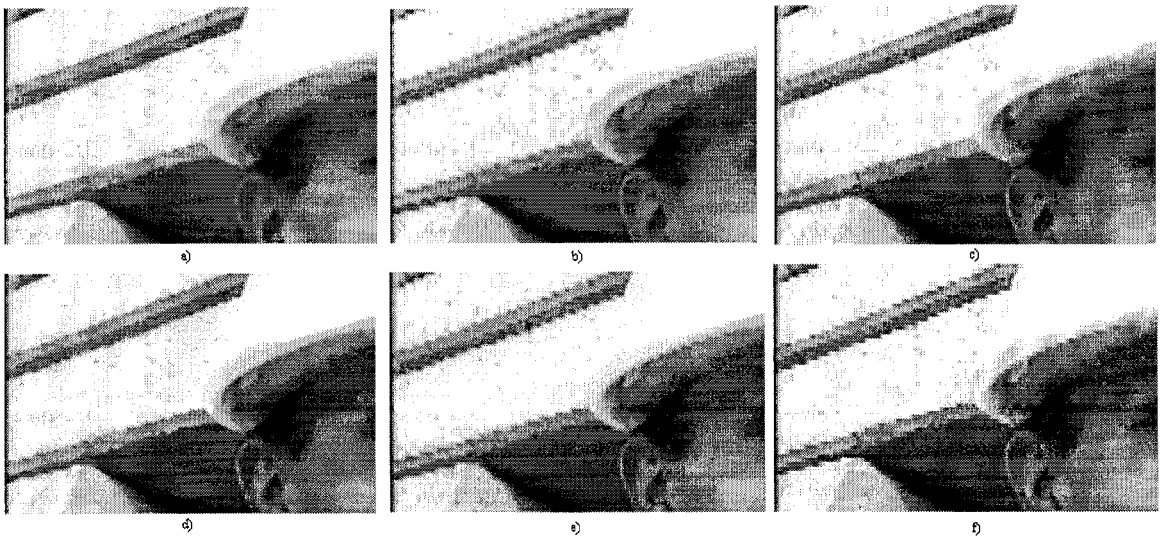


Figure 4.15. Frame from "Foreman" sequence deinterlaced by a) PBDI b) DOI
c) Enhanced ELA d) ELA e) Line Averaging f)Line Repetition

As mentioned, MSE at the core of PSNR is often criticized for its poor correlation with the human visual system [1][14], but because of a lack of a well-accepted alternative, it is still widely used as a criteria for objective evaluation. This conundrum is illustrated for the "Foreman" sequence by the relative performances of PBDI and DOI shown in Table 4.1 and Figure 4.15. In Table 4.1, DOI has the best PSNR, whereas in Figure 4.15, PBDI obviously performs better.

To further illustrate this point, we calculated the PSNR in a set of regions of interest with particularly well defined edges defined by the union of the internal parts of the polygons shown in Figure 4.16. The results of Table 4.2 show the improvement achieved by PBDI over the other methods. The PSNR is improved by 0.41 dB to1.55 dB in the set of selected regions of interest.

Figure 4.16. Edge areas in Foreman picture

TABLE 4.2. PSNR COMPARISON IN THE REGION OF INTEREST (IN DB)

	Line Rep.	Line Ave.	ELA	Enh. ELA	DOI	PBDI
Foreman (region of interest)	20.67	24.19	24.58	25.33	24.83	25.74
PBDI Improvement	5.07	1.55	1.16	0.41	0.91	

4.4.2. Computational Complexity

An important performance metric is computational complexity, especially when considering real-time implementation in hand-held devices.

For a given missing pixel, the PBDI algorithm performs twenty-one pattern comparisons to determine the edge direction. Away from the image borders, twelve of these comparisons have been calculated for previous pixels, meaning that only nine new comparisons are required for each pixel. A pattern comparison consists of two parts: the SAD and GBD. The SAD calculation requires three absolute differences and two additions. The GBD requires four differences, two absolute differences and one addition. Assuming $\alpha = 0.5$, the final error calculation requires two shifts and one addition.

Twenty comparisons are required to select the matching patterns, and four to calculate the median. To calculate the directional interpolation, the average of four (or two) pixels is needed, requiring three (or one) additions and one shift. Furthermore, to calculate the GBD in all patterns comparisons, the difference of every two sequential pixels can be saved in the upper and lower lines.

A 16-byte buffer can save all pixel differences for the current pixel. Consequently, four fewer difference calculations in each pattern comparison and only two more

difference instructions for each new pixel are needed. Buffers of 12 integers and 16 bytes can save the previous pattern comparisons and every two sequential pixel differences respectively. Using these buffers, the total number of operations decreases from 448 to 163.

The case of DOI can be analyzed in a similar way. This method uses eleven pixels in each line to calculate the missing pixel in the search range [-4, 4]. For this case, DOI uses eighteen pattern comparisons. Each one requires six absolute differences, five additions and six square instructions. Sixteen pattern comparisons are needed to select the best matching patterns. None of them can be reused directly, but four pixels comparisons have already been calculated for previous patterns. Furthermore, DOI needs one addition, one absolute value and one comparison to verify if the upper and lower patterns are aligned. It also needs three additions and three shifts to calculate the directional interpolation. Using buffers of 8 integers can reduce the total number of operations from 439 to 225.

A similar analysis was performed for the simpler deinterlacing methods, and the results are combined in Table 4.3. With the use of data buffers, the proposed PBDI algorithm has a computational complexity that is approximately 27% less than the DOI method. Moreover, DOI requires a three-line buffer while PBDI requires a single line.

TABLE 4.3. INSTRUCTION COUNT FOR DIFFERENT METHODS

Deinterlacing methods	Add, Sub, Com.	Abs.	Shift	Square	Total
ELA	6	3	1		10
EELA	19	7			27
DOI	219	109	3	108	439
PBDI	261	63	45		448
DOI (Data Buffered)	147	37	2	36	225
PBDI (Data Buffered)	116	45	21		163

4.5. Conclusion

A new edge-based deinterlacing method was proposed. It uses pattern comparison between upper and lower lines to achieve reliable edge direction. Twenty-one edges in nine different directions are considered. A gradient-based criterion is combined with SAD to achieve more reliable pattern matching. The proposed PBDI produces clearer and sharper edges when compared with conventional edge-based methods. Objective comparisons also show that the PBDI compares favorably to previous approaches.

4.6. References

[1] G. De Haan and E. B. Bellers, "Deinterlacing – An Overview," Proc. of IEEE, vol. 86, no. 9, Sep. 1998, pp. 1839-1857.

[2] T. Doyle, "Interlaced to sequential conversion for EDTV applications," in Proc. 2nd Int. Workshop Signal Processing of HDTV, Feb. 1988, pp. 412-430.

[3] H. Y. Lee, J. W. Park, T. M. Bae, S. U. Choi, and Y. Ho Ha, "Adaptive scan rate up-conversion system based on human visual characteristics," IEEE Transactions on Consumer Electronics, vol. 46, Nov. 2000, pp. 999-1006.

[4] S.-F. Lin, Y.-L. Chang, and L.-G. Chen, "Motion adaptive interpolation with horizontal motion detection for deinterlacing," IEEE Transactions on Consumer Electronics, vol. 49, No. 4, Nov. 2003, pp. 1256-1265.

[5] G. de Haan and R. Lodder, "De-interlacing of video data using motion vectors and edge information," Digest of ICCE, Jun. 2002, pp. 70-71.

[6] H. Yoo and J. Jeong, "Direction-oriented interpolation and its application to de-interlacing," IEEE Transactions on Consumer Electronics. Vol. 48, No. 4, November 2002, pp. 954-962.

[7] Y. L. Chang, S. F. Lin, and L. G. Chen, "Extended intelligent edge-based line average with its implementation and test method," Proceedings of ISCAS, 2004, pp. II 341-II 344.

[8] Y. L. Chang, S. F. Lin, C. Y. Chen, and L. G. Chen, "Video De-Interlacing by Adaptive 4-Field Global/Local Motion Compensated Approach," IEEE Transactions on Circuits and Systems for Video Technology, Vol. 15, No. 12, December 2005, pp. 1569-1582.

[9] H. -S. Oh, Y. Kim, Y. -Y. Jung, A. W. Morales, and S. -J. Ko, "Spatio-temporal edge-based median filtering for deinterlacing," IEEE International Conference on Consumer Electronics, 2000, pp. 52-53.

[10] G. Jeon, M. Anisetti, V. Bellandi, E. Damiani, and J. Jeong, "Fuzzy Weighted Approach to Improve Visual Quality of Edge-Based Filtering," IEEE Transactions on Consumer Electronics. Vol. 53, No. 4, Nov. 2007, pp. 1661-1667.

[11] C. Ballester, M. Bertalmío, V. Caselles, L. Garrido, A. Marques, and F. Ranchin, "An Inpainting- Based Deinterlacing Method," IEEE Transactions on Image Processing. Vol. 16, No. 10, Oct. 2007, pp. 2476-2491

[12] H. Mahvash Mohammadi, J.M.P. Langlois and Y. Savaria, " A Five-Field Motion Compensated Deinterlacing Method Based on Vertical Motion," IEEE Trans. on Consumer Electronics, vol. 53, no. 3, August 2007, pp. 1117-1124.

[13] A. M. Tekalp, Digital Video Processing. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[14] Y.-L. Chan, W.-C. Siu, "Reliable block motion estimation through the confidence measure of error surface," Signal Processing (76), No. 2, 1 July 1999, pp. 135-146.

[15] http://www.grm.polymtl.ca/~savaria/Deinterlaced_sequences/PBDI/

Chapter 5

Hybrid Video Deinterlacing Algorithm Using Reverse Motion Estimation

Hossein Mahvash Mohammadi, Yvon Savaria, Fellow, IEEE and J.M. Pierre Langlois, Member, IEEE

Abstract— *This paper proposes a new hybrid video deinterlacing algorithm method featuring a novel approach to qualify the reliability of motion vectors. The algorithm switches between motion compensated and line averaging methods based on motion vector reliability. When the motion vectors are calculated, reverse motion estimation is applied to the optimal matching block. A motion vector is assumed reliable if the result of reverse motion estimation refers to the original block or to a block in its vicinity. Motion compensation is used when motion vectors are reliable to improve the vertical resolution and line averaging is used when the motion vectors are not reliable to prevent artifacts. Experimental results show that reverse motion estimation performs better than previous approaches, based on objective and subjective criteria. The computational complexity of the proposed method is up to two orders of magnitude less than previous methods.*

Index Terms— Deinterlacing, Motion compensation, Reverse motion estimation, Edge-based methods, Directional Interpolation.

5.1. Introduction

Video interlacing was originally proposed to decrease the transmission bandwidth. It

is not the best solution to reduce bandwidth, but compatibility with current standards maintains its necessity in the very large TV market [1]. Deinterlacing is necessary to convert existing interlaced standards to EDTV and HDTV formats. HDTV supports both interlaced and progressive scans with up to 1080 scan lines.

Deinterlacing methods are generally classified into four classes: intra field, inter field, motion adaptive and motion compensated methods. Intra field deinterlacing algorithms use spatial interpolation techniques to calculate the missing pixels. These methods work well in video sequences with low spatial frequency, but they produce annoying flickering artifacts in areas with high spatial frequencies. Inter field deinterlacing algorithms exploit temporal correlation and apply temporal interpolation techniques. They improve vertical resolution in still areas but produce combing artifacts in moving areas.

Motion adaptive methods switch between intra and inter field techniques to take advantage of both methods in low spatial frequency and still areas. A motion detection algorithm is needed to discriminate between moving and still areas. Motion adaptive methods often produce flickering artifacts in moving areas with high spatial frequencies.

Motion compensated methods aim to virtually remove video data motion and take advantage of inter field techniques in both still and moving areas. Motion estimation is necessary to calculate the displacement of blocks or objects within neighboring fields. The quality of motion compensated video processing methods highly depends on the reliability of calculated motion vectors [2].

Hybrid deinterlacing methods combine elements from two or more classes. Several

works have combined motion compensated and intra field methods based on the reliability of calculated motion vectors [3]-[6].

This paper introduces a new method to qualify motion vector reliability. The method is called "reverse motion estimation" (RME). It is applied in a hybrid motion compensated algorithm (HMC) in which the five field motion compensated algorithm (FFMC) [7] is combined with line averaging, a simple intra field deinterlacing algorithm.

The paper is organized as follows. Section 5.2 presents an overview of motion compensated methods, FFMC and motion vector reliability measures. Section 5.3 introduces the RME measure and the HMC method. Section 5.4 compares the effectiveness of the different reliability measures when they are used in HMC methods. The comparisons are done in terms of objective and subjective evaluations of video quality and on computational complexity. Section 0 concludes the paper.

5.2. Background

5.2.1. Motion Estimation

Block matching is the most common and popular method for motion estimation. It consists of splitting an image into blocks, usually of size 8×8 or 16×16. For each block in the current field, a motion estimator searches neighboring fields to find the most similar block. It is often called the "optimal matching block" (OMB). The displacement between a block and its OMB is called a motion vector.

The effectiveness of motion compensation highly depends on the accuracy of motion

estimation. Four main factors affect this accuracy.

The first factor concerns the selection of a search method. Many different search methods have been proposed in the literature. The full search method is the most accurate but most costly search method. Fast search methods [8]-[11] aim to decrease the number of comparisons necessary to find the OMB. They rely on the assumption that as the distance from the OMB decreases the block comparison error also tends to decrease. This is not always true and fast search methods can get trapped in local minima.

The second factor deals with the criteria to assess block similarity. Some choices include the mean absolute difference (MAD), the sum of absolute differences (SAD) and phase correlation [13]. The MAD and SAD are fairly simple to calculate. Phase correlation is based on the Fourier transform of blocks and can provide better results than the MAD and SAD, but it involves much greater computational complexity.

The third factor concerns how the search is conducted in time. Forward and backward motion estimation algorithms search for OMBs in the previous or following fields, respectively. Bidirectional motion estimation uses both directions, and it can solve the problems of object occlusion, appearance and disappearance.

Alternation of existing lines to odd and even lines in interlaced sequences is another factor which complicates motion estimation [7]. Most methods use pre-filtering with an intra-field method prior to motion estimation to calculate missing pixels. Other methods use only same parity fields for motion estimation in which the existing lines are in the same position and pre-filtering is not necessary. This can produce artifacts in case of very fast and non-uniform motion.

5.2.2. FFMC

The FFMC algorithm uses two previous and two next fields for forward and backward motion estimation [7]. Pre-filtering by line averaging is used for the current and two next fields prior to motion estimation. The motion estimator finds the most similar block in each reference field. The block with the smallest error amongst all fields is selected as the OMB.

Motion compensation in FFMC is based on the amount of vertical motion and the reference field in which the final OMB was found. If the OMB is found in the same parity field and the amount of vertical motion per field is an even number, the block to which a half motion vector points in the opposite parity field is used to calculate missing pixels. In this case, the fact that the block and its OMB have the same missing and existing pixels makes motion estimation much more reliable. The missing pixels of the block exist in the opposite parity field at the position to which a half motion vector points. In such a case, vertical resolution improvement can be achieved if the motion is uniform and not too fast.

Using data from a block with half motion vector in the opposite parity field can produce artifacts. To prevent this, the method compares the error of the OMB with the error of the block obtained with a half motion vector. The error of the block must be smaller than the OMB error multiplied by a factor determined empirically.

If the OMB is found in the opposite parity field or if the amount of vertical motion per field is an odd number, the OMB is directly used to calculate the missing pixels.

5.2.3. Unreliable Motion Vectors

Vertical resolution improvement hinges on the availability of reliable motion estimation. Unreliable motion vectors can produce annoying artifacts. Several circumstances increase the production of unreliable motion vectors.

Conventional motion estimators compute a single motion vector for each block, assuming that the block does not change shape between fields. Zooming, rotation and object deformation are examples of affine transformations where this assumption is not true. In such cases, the true motion differs for each pixel in a block. Unfortunately, deformable block matching motion estimation with affine transformation is not practical because of its computational complexity [13].

When an object moves outside of the search window, the motion estimator calculates a wrong motion vector. Increasing the size of the search window resolves this problem but it drastically increases the computational complexity, especially if the full search method is used.

Pre-filtering performed prior to motion estimation can also affect motion vector accuracy. In this case, motion vectors are calculated based on original and estimated pixels.

Consequently, even the most powerful motion estimation algorithms may fail to calculate reliable motion vectors. In general, a motion estimator that can provide reliable motion vectors in all situations either involves too much computational complexity, is not practical or does not exist.

5.2.4. Estimating Motion Vector Reliability

Estimating the reliability of motion vectors is necessary for video processing applications using motion estimation and compensation. It is also a key element in HMC deinterlacing methods to achieve maximum vertical resolution improvement while preventing potentially annoying artifacts.

Different measures of motion vector reliability have been proposed in the literature. These methods exploit the OMB error, image statistic or motion continuity and smoothness.

The simplest methods use the OMB error or displaced frame difference (DFD) to qualify a motion vector [4][13]-[14]. DFD is calculated as follows:

$$DFD=\sum_{x\in\Lambda}\left|f_k(x)-f_m(x+d)\right| \qquad (5.1)$$

where f_k is the current frame, f_m is the reference frame, x refers to the spatial coordinates of a pixel, Λ is the set of spatial positions belonging to a block, and d is the motion vector candidate. A small value of DFD implies a more reliable motion vector. DFD tends to be high in areas with high spatial frequency and small in uniform areas, whether motion vectors are reliable or not. Consequently, it is difficult to pick a single good threshold to discriminate between reliable and unreliable motion vectors in all situations. Normalizing DFD by block intensity gradients can sometimes solve this problem [15], but not around straight edges or in occluded areas.

A different approach, Local Vector Smoothness (LVS), is based on motion vector continuity [16]. It is calculated by summing the absolute differences between the current

block's motion vector and the motion vectors of eight other blocks. These blocks overlap the current block and are shifted by one pixel in eight directions:

$$\frac{1}{LVS(x_b, y_b)} = \sum_{i=x_b-1}^{x_b+1} \sum_{j=y_b-1}^{y_b+1} \left\| d_{ij} - d_{x_b y_b} \right\| \qquad (5.2)$$

In (5.2), d is a motion vector and x_b and y_b are the current block's horizontal and vertical coordinates. When the motion vectors in a given area are similar, it is assumed that they are more reliable and the LVS metric becomes very large. This assumption is not true for blocks that are on or close to the boundary of a moving object.

Wang et al. have considered the a posteriori probability of motion vectors as a measure for motion vector reliability [3]. It is calculated as follows:

$$P(d \mid f_{k-1}, f_k) \approx \frac{1}{C} \exp\{-U(d \mid f_{k-1}) - \frac{1}{2\sigma^2} \sum_{x \in \Lambda} [f_k(x) - f_{k-1}(x+d)]^2\} \qquad (5.3)$$

where C is a constant, σ^2 is the variance of the displaced pixel differences $f_k(x) - f_{k-1}(x+d)$ and $U(.)$ is a energy function to impose local smoothness constraints on the variation of motion vectors. The function $U(.)$ is defined by the sum of the squared differences of motion vectors over a neighborhood calculated by:

$$U(d \mid f_{k-1}) = \frac{1}{2\sigma_d^2} \min\left\{ \left\| d - d_i \right\|^2 \mid 0 < i \le 4 \right\} \qquad (5.4)$$

where i refers to the four neighboring motion vectors of the blocks in the upper, lower, left and right sides, and σ_d^2 is the variance of the difference $d - d_i$. The constant C in (5.3) is calculated by:

$$C = \sum_{\xi \in D} \exp\{-U(\xi \mid f_{k-1}) - \frac{1}{2\sigma^2} \sum_{x \in \Lambda} [f_k(x) - f_{k-1}(x+\xi)]^2\} \qquad (5.5)$$

where D is the set of all possible estimates for the true motion vector d. The constant C involves much computational complexity, because it should be calculated over all possible estimates. To reduce this calculation cost, a simplification was made. The true motion vector estimates are assumed to be within a distance of ±2 pixels from the calculated motion vector d, or equal to the motion vectors of the neighboring blocks in up, down, left and right sides of the block. This assumption can significantly affect the method's accuracy, especially in sequences with fast motion where the motion vector can lie outside area D.

5.3. Proposed hybrid motion compensated algorithm using reverse motion estimation

5.3.1. Reverse Motion Estimation

The proposed RME method uses a straightforward technique to recognize unreliable motion vectors. The technique is based on the assumption that, when motion vectors are reliable, the OMB relationship is reciprocal between blocks in different fields. The motion estimator first finds a candidate OMB. It then searches within the field containing the original block to find the block that is most similar to that OMB. If this most similar block is the original block or a block close to it, then the motion vector is assumed to be reliable.

The RME method is illustrated in Figure 5.1. A reliable motion vector is sought for

block *A* in field *n* to some block in field *n+1*. Block *B* in field *n+1* is found as the OMB for block *A*. Reverse motion estimation is performed, and block *C* in field *n* is found as the best match for block *B*. The Euclidian distance between blocks of *A* and *C* qualifies the motion vector. Its value is between zero and the search window size. If it is less than a user-defined threshold, the motion vector is assumed reliable.

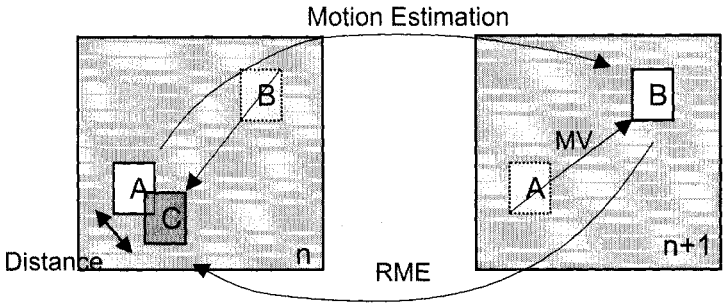


Figure 5.1. RME

5.3.2. Hybrid Motion Compensated Algorithm

The enhanced hybrid motion compensated deinterlacing algorithm proposed in this paper is shown in Figure 5.2. It switches between the FFMC and line averaging methods according to motion vector reliability, as estimated by RME.

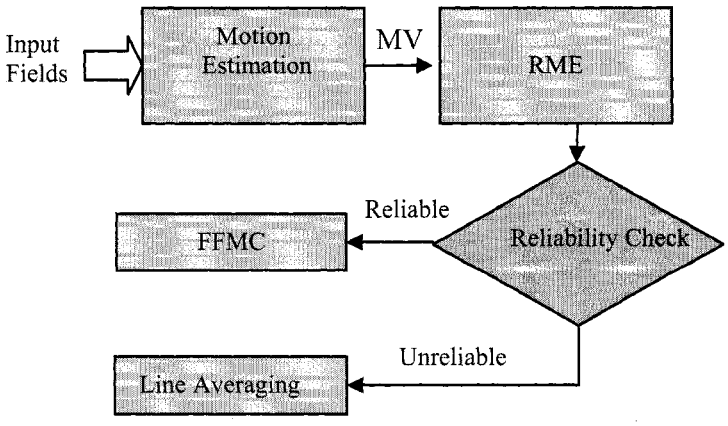


Figure 5.2. Structure of the proposed HMC method

Line averaging is used when motion vectors are declared unreliable. Although several intra field methods have better performance, it is very simple to implement and being uniform in the other steps of the algorithm facilitates the comparison of RME with other motion vector reliability estimating methods. In a complete deinterlacing method, line averaging would advantageously be replaced by intra field methods such as PBDI [17] or DOI [18].

Figure 5.3 shows a detailed flow chart of the HMC method with RME. An initial pre-filtering step with line averaging is applied in fields n to $n + 2$ to obtain missing lines prior to motion estimation. Fields $n - 2$ and $n - 1$ have already been deinterlaced. A standard motion estimator searches fields $n - 2$ to $n + 2$ to find the OMB. The associated motion vector is annotated d_1 and is found in field m. Field n is then searched to find the best match for the OMB, called OMBRME with motion vector d_2. If the distance $||d_1 - d_2||$ between the original block and block OMBRME is larger than the distance threshold *DisThr*, motion vector d_1 is assumed to be unreliable.

If the motion vector is reliable, FFMC calculates the missing pixels. Otherwise, no action is necessary and the results of the pre-filtering step are kept.

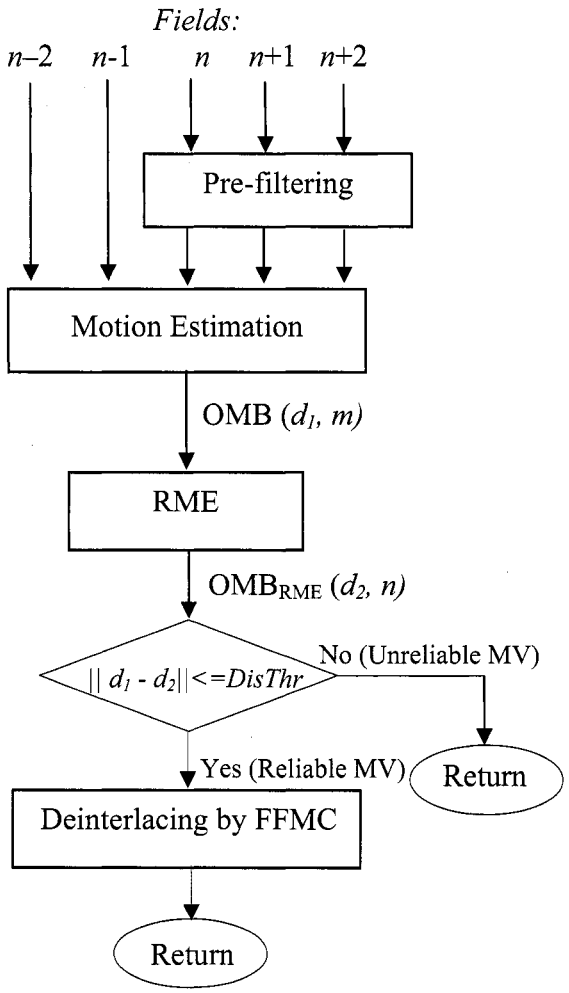


Figure 5.3. HMC using RME

5.4. Experimental Results and Analysis

5.4.1. Test methodology

Fifteen progressive color sequences were first interlaced. They were then deinterlaced three times by the HMC algorithm shown in Figure 5.2, using RME, LVS

[15] and a posteriori [3] methods to provide motion vector reliability estimates. In each case, the specific features of each method required a detailed exploration of the threshold space as each of these methods is controlled by thresholds that must be set appropriately for best results. The a posteriori method is based on a conditional probability, LVS measures neighboring motion vector smoothness, and RME is based on motion vector reciprocity. The three reliability methods calculate heterogeneous values requiring different thresholds to recognize reliable motion vectors. For example, the a posteriori method produces a metric in the interval [0, 1], while LVS and RME can produce values much greater than 1.

Finding an optimum threshold is always a challenging issue in HMC methods. Overestimating the reliability often results in visual artifacts while underestimating it increases flickering in areas with high spatial frequencies. It also prevents vertical resolution improvement that motion compensated processing can provide. In order to overcome this problem and to find the optimum threshold value resulting in the minimum overall error, multiple simulations were performed using different threshold values. The thresholds were varied in the intervals [0, 10], [0, 10] and [0, 1] for the RME, LVS and a posteriori methods, respectively. A final difference is obtained by noticing that high scores imply reliability for the a posteriori method but unreliability for RME and LVS.

Motion estimation was performed only on the luminance component "Y". A block size of 8 × 8, a search window size of 24 × 24 and the full search method were used in all cases. Deinterlacing was applied to the three RGB color components, whether by motion compensation or line averaging.

The Mean Squared Error (MSE) between the deinterlaced and original color sequences was used for objective evaluation. The MSE is not a very sensitive measure to compare image processing algorithms [3]. In the present case, an artifact caused by an erroneous motion vector has a significant visual impact on subjective evaluation, but it may increase the MSE by only a small amount. Still, the MSE remains in wide use as an objective criterion because of a lack of a well-accepted alternative.

In order to objectively assess the number of artifacts produced by each method, we propose a new metric based on block error statistics. Our method is based on the observation developed in subjective tests that annoying artifacts due to inaccurate motion estimation tend to produce large localized block errors. We therefore measured the error for all individual blocks for each method, and produced a cumulative frequency analysis used as an objective measure of artifact counts.

5.4.2. Test data analysis and objective evaluation

Table 5.1 reports the MSE over 50 frames of fifteen sequences deinterlaced by line averaging alone, FFMC alone and the three variations of HMC. For HMC, results are reported for three different threshold values. In the table, bold type indicates a result where HMC improves performance over FFMC alone.

The sequences in Table 5.1 are categorized into three different types and ordered according to the amount of motion and number of situations that may cause unreliable motion vectors. Overall results for each part and for all sequences are also reported.

TABLE 5.1. MSE COMPARISONS IN HMC METHODS BY DIFFERENT MOTION VECTOR RELIABILITY MEASURES

Seq. Name	Motion	LA	FFMC	LVS [15]			A posteriori [3]			Proposed RME		
				T=0	T=5	T=10	T=1	T=0.5	T=0.1	T=0	T=1	T=5
Claire_C	low	10	2.7	2.8	**2.6**	2.7	2.7	2.7	2.7	2.7	2.7	2.7
Grandma		30	3.1	3.2	**2.9**	**2.9**	3.1	3.0	3.0	3.0	3.0	3.0
MotherD.		30	9.3	11	8.8	9.0	10.6	9.1	9.2	**8.7**	**8.7**	9.1
Salesman		46	6.0	7.0	**6.0**	**6.0**	6.9	6.2	6.2	6.1	**6.0**	**6.0**
Part 1 overall		116	21.1	24	**20.3**	20.6	23.3	21	21.1	20.5	20.4	20.8
Foreman	mo dest	50	**35**	38	36	**35**	40	38	37	36	36	**35**
Hall M.		72	16	24	**15**	**15**	28	18	17	17	16	**15**
Highway		48	43	**42**	**42**	**42**	**42**	**42**	43	**42**	**42**	**42**
News		73	12	8.5	**6.4**	6.6	12	12	12	6.7	6.6	6.8
Silent		43	**6**	15	8.8	7.6	17	8.7	7.5	8.4	7.6	6.9
Flower		640	174	252	181	176	279	194	190	188	174	**173**
Mobile		669	337	413	351	339	434	356	350	356	337	**335**
Part 2 overall		1595	623	793	640	621	852	669	657	654	619	**614**
Commercia	hig h	564	369	384	365	**364**	411	365	**364**	366	366	365
Baseball		308	154	149	148	150	**146**	148	149	**146**	149	152
TableT.		155	94	94	91	92	96	91	91	91	**90**	91
Football		125	153	**107**	118	129	105	124	133	114	126	138
Part 3 overall		1152	770	734	722	735	758	728	737	**717**	731	746
Overall		2863	1414	1550	1383	1378	1634	1418	1414	1392	**1372**	1381

For the sequences in the first part, HMC methods should produce results very close to FFMC. Little or no improvements are expected, because these sequences contain very little movement. For the sequences in the second part, little improvement is expected, because only a few blocks may have unreliable motion vectors and the MSE is calculated as an average over all blocks. It is expected that HMC methods improve the results for the sequences of the third part of the table. Although, in general, line averaging does not provide better results than FFMC, it should perform better for blocks with incorrect motion vectors.

As shown in Table 5.1, most HMC results in the first part are almost equal to the FFMC results, as expected. The difference in overall MSE between the FFMC and HMC

methods in this part of the dataset is negligible. Most results of the second part show very little improvement compared with FFMC, and there is even some deterioration. RME often performs better than the other variations in this part, especially for sequences with high spatial frequencies such as "Flower" and "Mobile".

The maximum overall MSE reduction, when comparing with FFMC, is achieved for sequences in the third part. The maximum improvement is achieved in the "Football" sequence by all HMC methods. The LVS, a posteriori and RME methods improve this sequence by 1.55 dB, 1.64 dB and 1.28 dB comparing with FFMC, respectively. The thresholds for which the HMC methods provide the maximum improvement on the most favorable benchmark do not provide the best overall results. The best overall result for LVS, a posteriori and RME are obtained with thresholds equal to 10, 0.1, and 1, respectively.

Figure 5.4 shows the average of overall MSE for the fifteen sequences of Table 5.1 for different thresholds. It can be seen that the a posteriori method does not improve the results for all threshold values. LVS has the most improvement for a threshold value of 10, but the corresponding reliability level is not sufficient to prevent the introduction of artifacts. A threshold value of 1 in RME is optimum and can remove the artifacts.

The cumulative block error frequency distribution for the 50 frames of the "Football" sequence is shown in Figure 5.5 for the three methods. Figure 5.6 shows the cumulative frequency distribution relative to RME. The RME method has less than half the number of blocks with a large error (more than 2500) than the a posteriori method, and less than a quarter than the number produced by the LVS method.

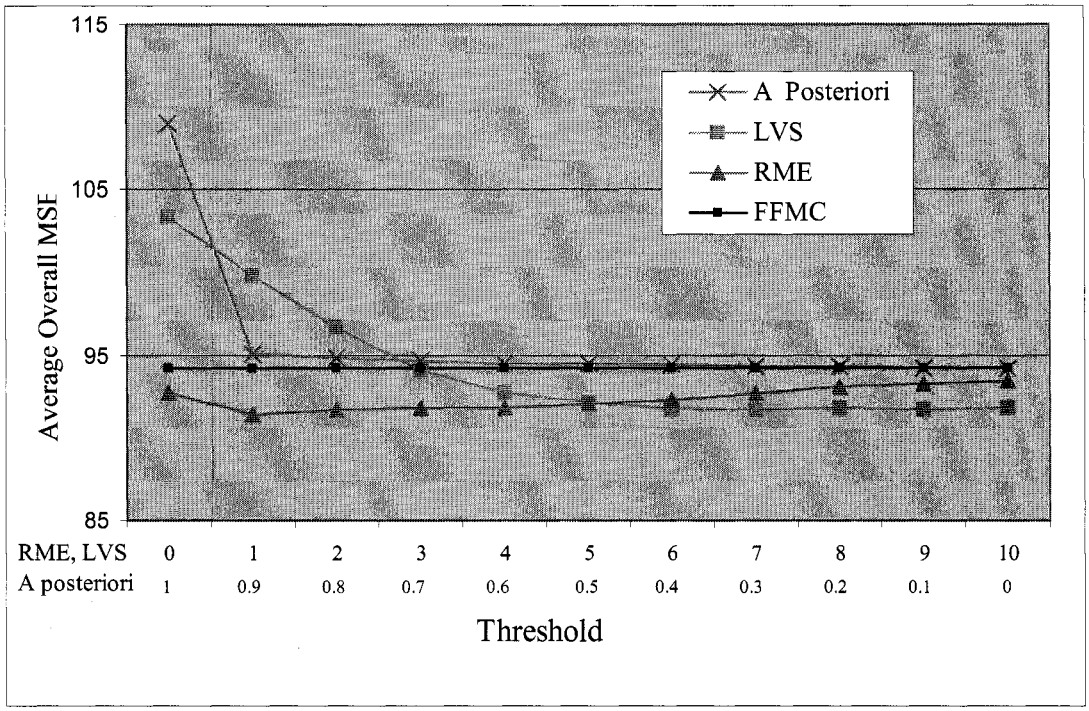


Figure 5.4. Average overall MSE in HMC methods using different measures and thresholds

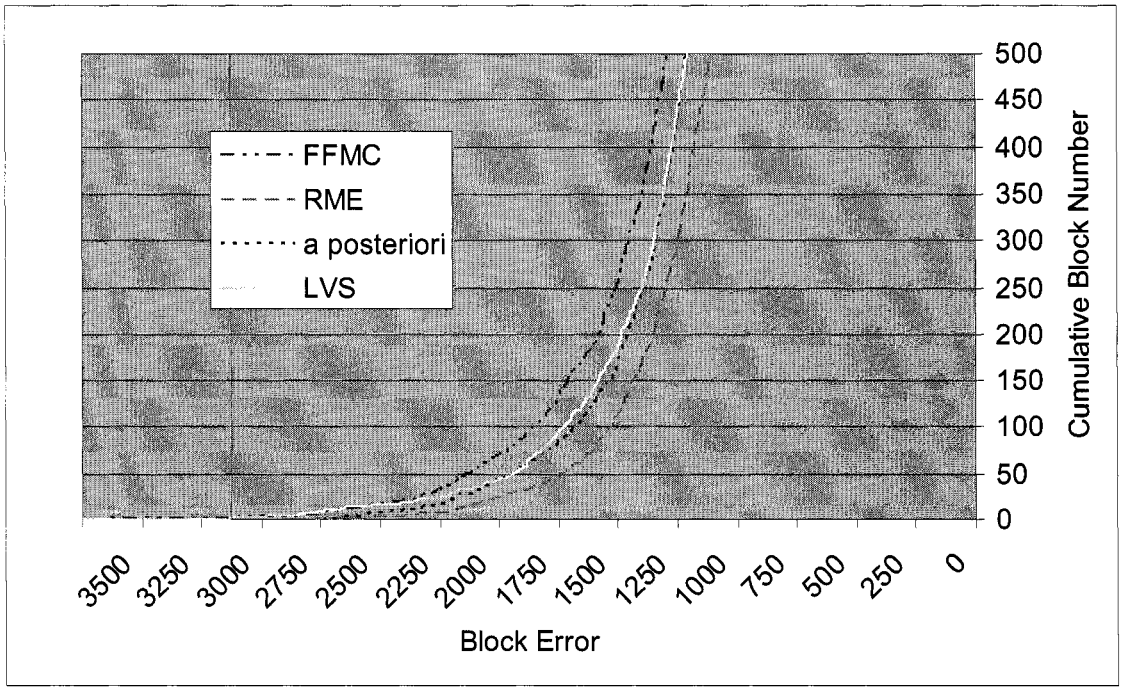


Figure 5.5. Cumulative frequency for block error for the "Football" sequence

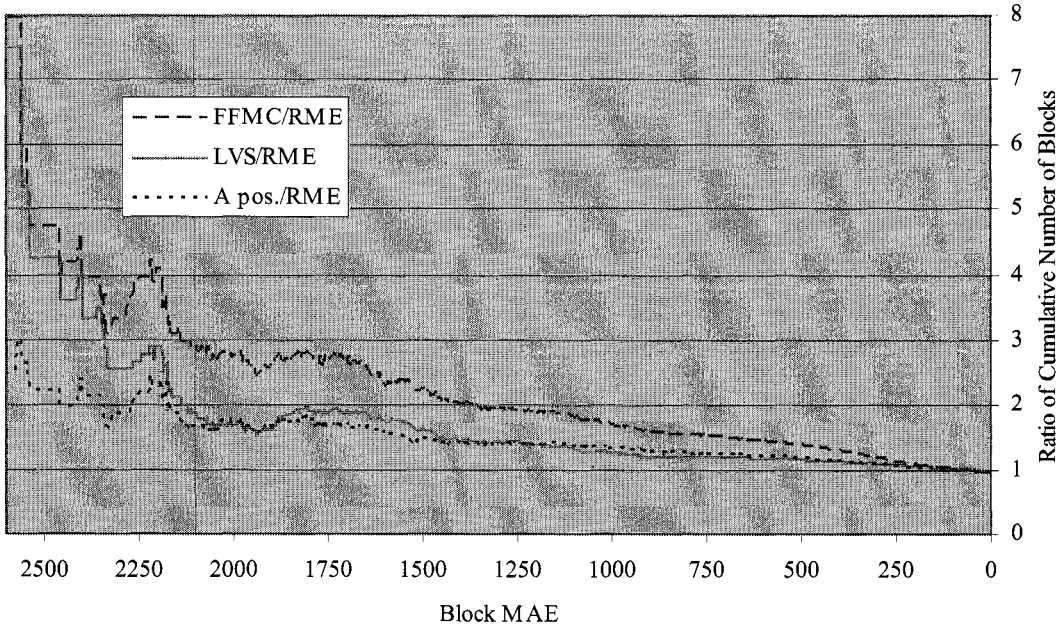
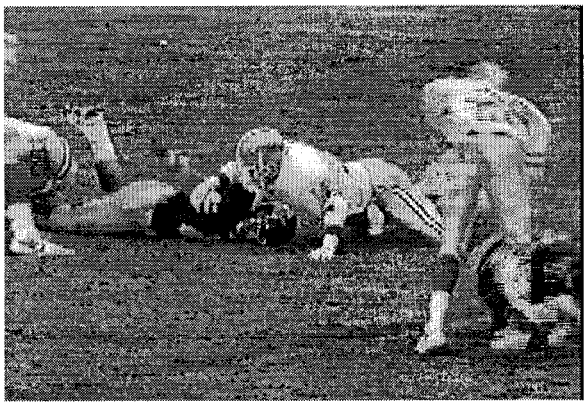


Figure 5.6. Cumulative frequency for block error relative to RME for the "Football" sequence

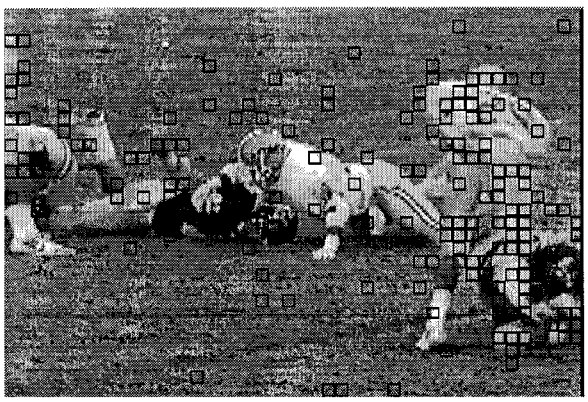
5.4.3. Subjective evaluation

The "Football" sequence was selected for subjective evaluation because of the intense motion it contains and the abundance of opportunities for assessing motion vector reliability. The HMC method was applied with the three variations of motion vector reliability assessment. For each case, the threshold was selected based on the minimum overall MSE as reported in Table 5.1. The thresholds for LVS, a posteriori and RME were thus equal to 10, 0.1 and 1, respectively. Figure 5.7 shows a frame of the football sequence deinterlaced by FFMC alone (5.7.a), HMC/RME (5.7.c), HMC/LVS (5.7.e) and HMC/a posteriori (5.7.g). Corresponding blocks with detected unreliable motion vectors are shown in figures 5.7.b, 5.7.d and 5.7.e The HMC/RME method recognizes blocks

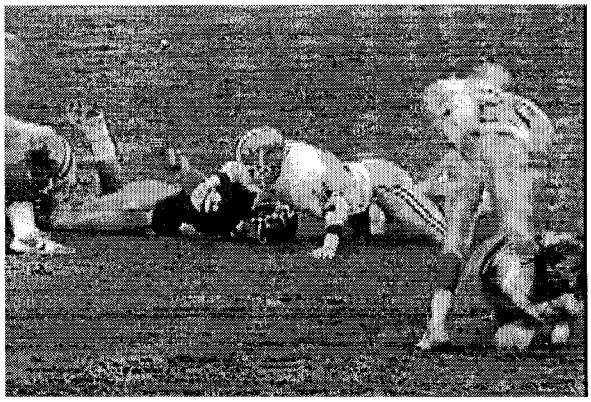
with unreliable motion vectors better than the other two variations. Most artifacts of the image deinterlaced by FFMC alone are removed in the image deinterlaced by RME, but that is not the case for the LVS and a posteriori methods. The optimum thresholds in the LVS and a posteriori methods require very low levels of reliability. Still, many blocks in the image background with no or little motion are recognized as blocks with unreliable motion vectors. On the contrary, the optimum threshold for the RME method is stringent but fewer background blocks are associated with unreliable motion.



a)



b)



c)

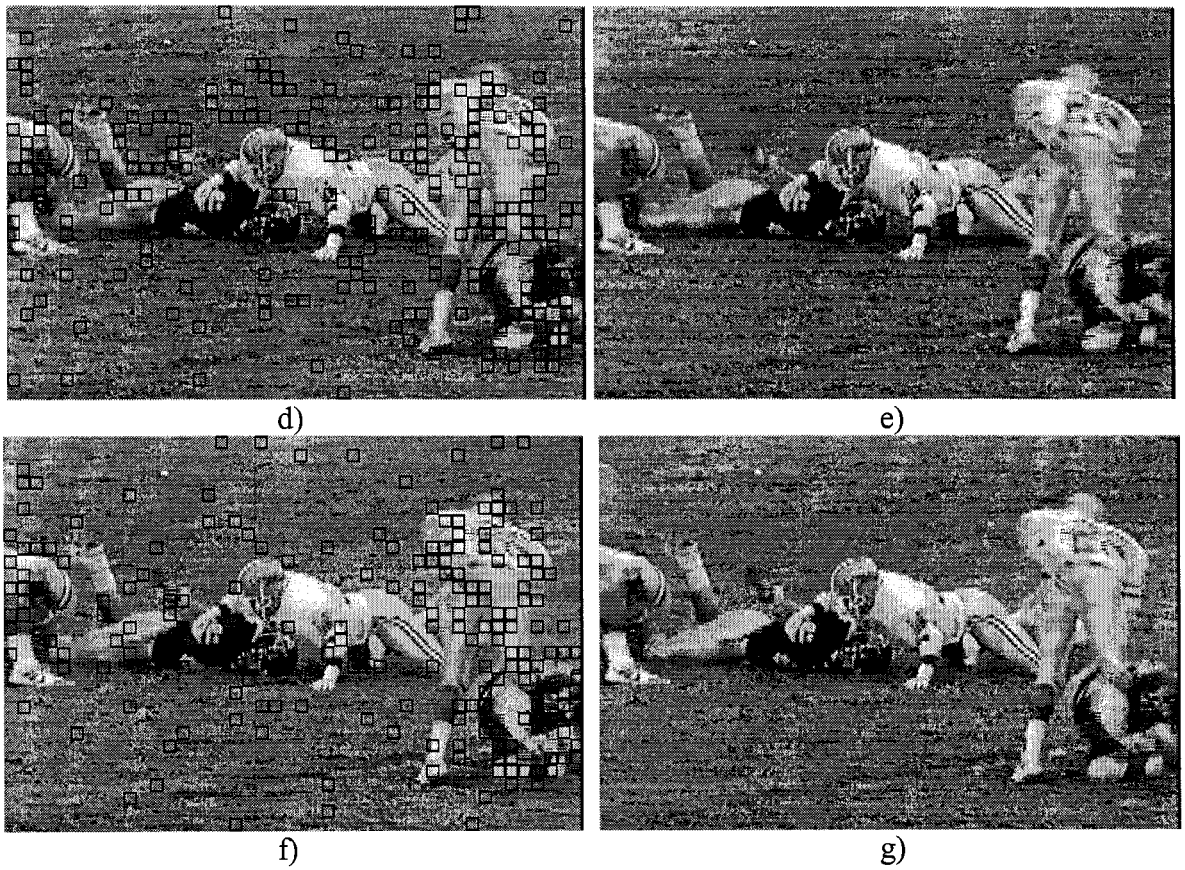


Figure 5.7. Subjective comparison a) FFMC b) Blocks with unreliable MVs by RME c) RME d) Blocks with unreliable MVs by LVS e)LVS f) Blocks with unreliable MVs by a posteriori g) A posteriori

In another experiment, we set the thresholds to maximum reliability in all three methods. The results for the "Flower-Garden" sequence show that the RME method produces less flickering artifacts than the LVS and a posteriori methods. This implies that the LVS and a posteriori methods force a switch to line averaging for more blocks than the RME method. We conclude that the LVS and a posteriori methods tend to underestimate the reliability of motion vectors. The results for this sequence and other experiments are available online [19].

5.4.4. Computational complexity

Motion estimation is the most complex part of motion compensated deinterlacing algorithms. The search method and number of frames used for motion estimation are two key issues for the complexity of the motion estimation. The FFMC method uses full search method and four reference fields to calculate a motion vector. The RME measure only searches one frame in the reverse direction to find an OMB and qualify a motion vector. The RME procedure is thus similar to what FFMC performs for motion estimation for one frame. Therefore the RME complexity is almost ¼ of the motion estimation performed by FFMC.

As shown in (5.2), the LVS method requires the motion vectors of eight additional blocks to qualify a motion vector. For each block, the procedure is to find an OMB and calculate a motion vector that is similar to RME method in term of execution time. Therefore the complexity of LVS is almost eight times that of the RME complexity and two times that of FFMC motion estimation. The a posteriori measure involves calculation of motion vectors, variance and other factors that makes the method very complex.

The relative computational complexities of the RME, LVS and a posteriori methods can be evaluated based on their execution time within the HMC algorithm. Table 5.2 shows the execution times for software implementations executed on a Pentium IV processor (1.7 GHz) in milliseconds. These results are single frames of various sizes. As shown in Figure 5.8, the additional execution time needed for of each motion vector reliability measure is a linear function of the image size. The execution time for each HMC method can be calculated by adding the execution times of FFMC to that of each

measure of motion vector reliability. Based on the overall results, it is observed that RME is 8.75 and 113 times faster than the LVS and a posteriori methods, respectively, while it provides better overall MSE results. The execution time of HMC with RME is 1.24 times that of the FFMC alone, while for the LVS and a posteriori methods the HMC execution times are 3 and 28.7 times longer than FFMC alone, respectively.

TABLE 5.2. EXECUTION TIME IN MILLISECONDS

Sequence	Image Size	FFMC	LVS	A posteriori	RME
Baseball	160×112	612	1181	15449	140
Commercial	160×112	627	1262	16279	144
Claire_c	176×144	848	1689	22342	201
Flower Garden	352×240	3096	6425	85479	670
CoastGuard	352×288	3650	7618	100016	934
Relative to FFMC		1.00	2.06	27.12	0.24

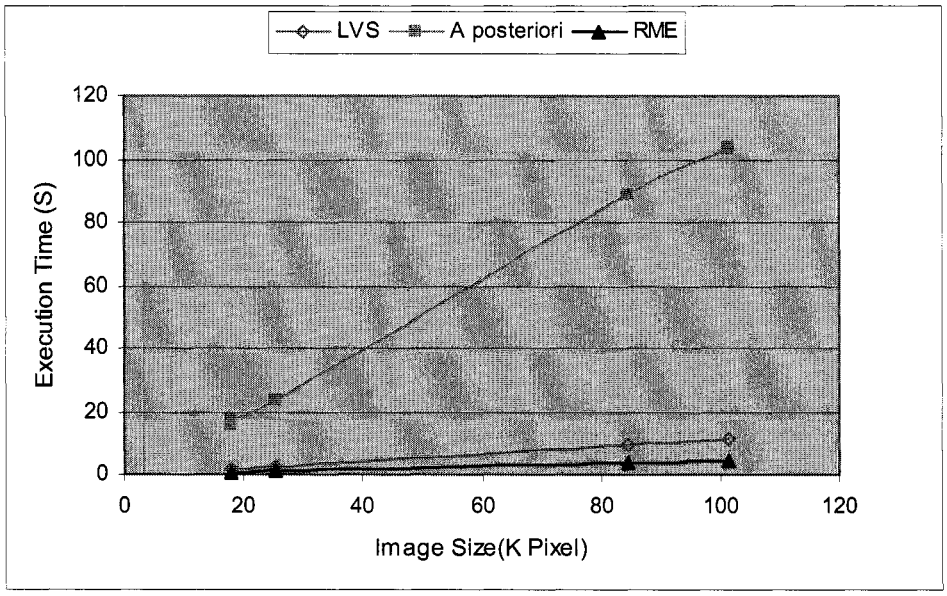


Figure 5.8: Execution time versus image size in HMC methods

5.5. Conclusion

A new measure of motion vector reliability was proposed in this paper. In simple terms, it tries reversing the best motion vector detected to confirm its validity. This measure was applied in a proposed HMC deinterlacing algorithm in which a switch is made between from motion compensation to line averaging for blocks with unreliable motion vectors. Experimental results show the proposed RME performs better than previous work based on overall MSE over different sequences. The proposed RME also has significantly less computational complexity than other existing measures. The effectiveness of the RME is also justified based on objective and subjective resulting image quality evaluations.

5.6. References

[1] G. De Haan and E. B. Bellers, "Deinterlacing – An Overview," Proc. of IEEE, vol. 86, no. 9, pp.1839 – 1857, Sep. 1998.

[2] Y.Y. Jung, B.T. Choi, Yung-Jun Park and Sung-Jea Ko, "An effective de-interlacing technique using motion compensated interpolation," IEEE Trans. on Consumer Electronics, vol. 46, no. 3, pp. 460-466, Aug. 2000.

[3] D. Wang, A. Vincent, and P. Blanchfield, "Hybrid De-Interlacing Algorithm Based on Motion Vector Reliability," IEEE Trans. Circuit and Systems for Video Technology, vol. 15, No. 8, Aug. 2005, pp. 1019-1025.

[4] H. Mahvash M., J.M. P. Langlois, and Y. Savaria, "A Threshold-Based De-Interlacing Algorithm Using Motion Compensation and Directional Interpolation," IEEE ICECS, Nice France, Dec. 2006.

[5] J. Kovacevic, R. J. Safranek, and E. M. Yeh, "Deinterlacing by successive approximation," IEEE Trans. Image Process., vol. 6, no. 2, pp. 339–344, Feb. 1997.

[6] K. Sugiyama and H. Nakamura, "A method of de-interlacing with motion compensation interpolation," IEEE Trans. Consum. Electron., vol. 45, no. 3, pp. 611–616, Aug. 1999.

[7] H. Mahvash Mohammadi, P. Langlois and Y. Savaria, "A Five-Field Motion Compensated Deinterlacing Method Based on Vertical Motion," IEEE Trans. On Consumer Electronics, vol. 53, No. 3, August 2007, pp 1117-1124.

[8] S. Zhu and K. K. Ma, "A new diamond search algorithm for fast block matching motion estimation," Proc. of Int. Conf. Information, Communications and Signal Processing, vol. 1, pp. 292-6, 1997.

[9] L.K. Liu and E. Feig, "A Block-Based Gradient Descent Search Algorithm for Block Motion Estimation in Video Coding," IEEE Trans. Circuit and Systems for Video Technology, vol. 6, No. 4, pp. 419-422, Aug. 1996.

[10] S. Zhu and K.K. Ma, "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation," IEEE Trans. Image Processing, vol. 9, No. 2, pp. 287-290, Feb. 2000.

[11] Xuan-Quang Banh and Yap-Peng Tan "Efficient Video Motion Estimation Using Dual-cross Search Algorithms", IEEE ISCAS 2005, pp. 5485-5488.

[12] A. M. Tekalp, Digital Video Processing. Englewood Cliffs, NJ: Prentice-Hall, 1995.

[13] S.-C. Han and J.W.Woods, "Adaptive coding of moving objects for very low bit rates," IEEE J. Sel. Areas Commun., vol. 16, no. 1, pp. 56–70, Jan. 1998.

[14] E. Francois, J.-F. Vial, and B. Chupeau, "Coding algorithm with region based motion compensation," IEEE Trans. Circuits Syst. Video Technol., vol. 7, no. 1, pp. 97–108, Feb. 1997.

[15] L. Hill and T. Vlachos, "Fast motion estimation using reliability weighted robust search," Electron. Lett., vol. 37, no. 7, pp. 418–420, 2001.

[16] O. A. Ojo and G. de Haan, "Robust motion-compensated video upconversion," IEEE Trans. Consum. Electron., vol. 43, no. 4, pp. 1045–1056, Nov. 1997.

[17] H. Mahvash Mohammadi, J.M.P. Langlois and Y. Savaria, "A Pattern-Based Directional Interpolation Deinterlacing Algorithm," IEEE Transactions on Consumer Electronics (Submitted).

[18] H. Yoo and J. Jeong, "Direction-oriented interpolation and its application to de-interlacing," IEEE Transactions on Consumer Electronics. Vol. 48, No. 4, November 2002, pp. 954-962.

[19] http://www.grm.polymtl.ca/~savaria/deinterlaced_sequences/

Chapter 6: Conclusion

6.1. Contributions

A hybrid motion compensated algorithm was proposed in this thesis. A novel motion compensated method called FFMC and a new edge-based algorithm called PBDI were proposed and combined in the hybrid method. The hybrid method switches between FFMC and PBDI based on the reliability of motion vectors to achieve the maximum vertical resolution while preventing the creation of artifacts from incorrect motion vectors. A new measure, RME, was also proposed to evaluate the reliability of motion vectors. RME was applied in the hybrid method to decide which method should be used.

Motion estimation is the most complex part of a motion compensation method. The search method, size of the search window and number of reference fields determine the computational complexity of the motion estimation. Fast search methods significantly decrease the computational complexity of motion estimation. However, they may fail to provide accurate motion vectors in some situations. The advantage of the proposed hybrid method is possibility of using fast search method while preventing artifacts of inaccurate motion vectors. The hybrid method does not improve the vertical resolution of blocks deinterlaced by intra field method but due to few numbers of these blocks in each field, the quality of the whole picture is not affected much by this issue.

We proposed a new motion estimation technique that uses forward and backward motion estimation within two previous and two next fields of the same and opposite

parity. Four OMBs are found in the reference fields but only one with the minimum error is used to calculate the motion vector. The motion compensation proposed exploits the amount of vertical motion to calculate the missing lines from the existing lines. This improves the vertical resolution and reduces flickering artifacts in deinterlaced sequences.

Full search method has the highest computational complexity but the fast search methods may fail to calculate accurate motion vector if SAD does not change monotonously in the search window. FFMC used full search method to disable impact of inaccurate search method and show how FFMC is adapted to interlaced video data.

A new method to recognize edge directions was proposed in PBDI. It finds the maximum correlation between patterns in the lines above and below a missing pixel. The method used only nine pixels in each of two reference lines, but can recognize twenty-one edges in nine different directions. A combination of a gradient-based criterion with SAD was also proposed to calculate the pattern correlation. PBDI improves the edge direction detection, the discrimination between actual and misleading edges, and produces clearer and sharper edges when compared with conventional edge-based methods.

Reverse motion estimation was proposed to measure the reliability of motion vectors. RME repeats motion estimation in a direction opposite to the calculated motion vector by replacing the original and reference fields. If the motion vector calculated by reverse motion estimation is the same as or close to the primary motion vector, but in the opposite direction, the primary motion vector is assumed to be reliable. The difference

between these two motion vectors is used to evaluate the reliability of primary motion vectors. Simulation results show that the proposed RME performs better than previous works based on overall MSE over different sequences while it has significantly less computational complexity than other existing measures.

6.2. Future work

In this section, we mention some open research avenues and suggest future work based on our studies and experiments on this thesis.

- How to determine an appropriate value for the coefficient of α in the FFMC method? The appropriate value of α changes in different sequences. On the other hand, overestimating α may not prevent the artifacts of fast and non-uniform motions and underestimating it increases the use of pre-deinterlaced or pre-filtered lines, causing flickering artifacts.
- Investigate the use of fast search methods in the motion estimation and compensation used in the FFMC algorithm because the full search method requires a high computational complexity that is difficult to implement in real time systems.
- Increase the number of reference fields to calculate the missing line from the existing lines in case of sub-pixel vertical movement between fields in the FFMC algorithm. Further reference fields are needed so

that the existing lines of the current field may reappear and the missing lines exist in a reference field in case of sub-pixel vertical movement.

- Split a block into four small blocks in case of large OMB error in the FFMC algorithm. This can improve the results when a part of a block is not associated with the motion vector.
- Investigate how to use the RME concept to qualify blocks with half amount of the motion vector instead of using SAD. This can eliminate the need for the α factor in the FFMC algorithm.
- How to use a combination of GBD and SAD for block matching used for motion estimation? In some situations, this combination may result in more accurate motion estimation.
- To combine block-based and pixel-based motion compensation in the FFMC algorithm. This can improve the results when a motion vector is not associated with all pixels in the block.
- How to use a variable length of the patterns in the PBDI algorithm? PBDI uses three-pixel patterns but the best pattern size may differ for different sequences thus choosing the appropriate size can improve the quality of the results.

Bibliography

[1] AUBERTIN P., MAHVASH MOHAMMADI H., SAVARIA Y. a and LANGLOIS P., "A High Performance ASIP Implementation of the PBDI Intra-Field Deinterlacing Method," *IEEE NEWCAS 2009* (Accepted)

[2] BALLESTER C., BERTALMÍO M., CASELLES V., GARRIDO L., MARQUES A., and RANCHIN F., "An Inpainting- Based Deinterlacing Method," IEEE Transactions on Image Processing. Vol. 16, No. 10, Oct. 2007, pp. 2476-2491

[3] BANH X. Q. and TAN Y. P. "Efficient Video Motion Estimation Using Dual-Cross Search Algorithms", *IEEE 2005*, pp. 5485-5488.

[4] BELLERS E.B. and HAAN G. de, "Advanced de-interlacing techniques," *Proc. ProRISC/IEEE Workshop on Circuits, Systems and Signal Processing*, Mierlo, The Netherlands, November 1996, pp. 1-13.

[5] BYUN S., BYUN J., and KIM G., "A Recursive Approach For De-Interlacing Using Improved ELA And Motion Compensation Based Bi-Directional BMA," *International Conference on Image Processing (ICIP)*, 2004, pp. 1679-1682.

[6] CHANG Y. L., WU P. H., LIN S. F., and Chen L. G., "Four Field Local Motion Compensated De-Interlacing," *ICASSP 2004*, pp. V253-V256.

[7] CHANG Y. L., LIN S. F., and CHEN L. G., "Extended Intelligent Edge-Based Line Average with Its Implementation and Test Method", *ISCAS 2004*, pp. II 341-II 344.

[8] CHANG Y. L., LIN S. F., CHEN C. Y., and CHEN L. G., Video De-Interlacing by Adaptive 4-Field Global/Local Motion Compensated Approach, *IEEE Transactions*

on circuits and Systems for Video Technology, Vol. 15, No. 12, December 2005, pp. 1569-1582.

[9] CHEN M. J., HUANG C. H. and HSU C. T., "Efficient De-interlacing Technique by Inter-field Information," *IEEE Transactions on Consumer Electronics*, Vol. 50, No. 4, November 2004, pp. 1202-1208

[10] CHEN T., WU H.R., and. YU Z.H, "An efficient edge line average interpolation algorithm for deinterlacing," *Proc. SPIE: Visual Communications and Image Processing*, Vol. 4067, 2000, pp. 1551-1558.

[11] CHONG T. S., AU O. C., CHAU W. S., CHAN T. W., "A De-interlacing algorithm based on texture Classification," *ICCE 2005*, pp. 445-446.

[12] CHONG T. S., AU O. C., CHAU W. S., CHAN T. W., "A Content Adaptive De-interlacing Algorithm," *ISCAS 2005*, pp. 4923-4926.

[13] DEAME J., " Motion Compensated De-Interlacing: The Key to the Digital Video Transition," *SMPTE 141st Technical Conference in NY*, November 19-22, 1999.

[14] DOYLE T., "Interlaced to Sequential Conversion for EDTV Applications," *in Proc. 2nd International Workshop Signal on Processing for HDTV*, February 1988, pp. 412-430.

[15] GAO X., GU J., LI J., "De-interlacing Algorithms Based on Motion Compensation," *IEEE Transactions on Consumer Electronics*, Vol. 51, No. 2, May 2005, pp. 589-599.

[16] GHANBARI M., "The Cross-Search Algorithm for Motion Estimation," *IEEE Transactions on Communication*, Vol. 38, No. 7, July 1990, pp. 950-953.

[17] HAN D., SHIN C. Y., CHOI S. J. and PARK J. S., "A Motion Adaptive 3-D De-Interlacing Algorithm Based On The Brightness Profile Pattern Difference", *IEEE* 1999, pp. 338-339.

[18] HAN D., SHIN C. Y., CHOI S. J., and PARK J. S., "A Motion Adaptive 3-D De-Interlacing Algorithm Based On The Brightness Profile Pattern Difference," *IEEE Transactions on Consumer Electronics,* Vol. 45, No. 3, August 1999 , pp.690-697.

[19] HAAN G. De, "Motion compensated de-interlacing, noise reduction, and picture rate conversion," *IEEE Transactions on Consumer. Electronics,* Vol. 45, No. 3, August 1999, pp. 617–624.

[20] HAAN G. de and LODDER R., "De-Interlacing Of Video Data Using Motion Vectors And Edge Information," *Digest of the ICCE'02,* June 2002, pp. 70-71.

[21] HAAN G. de and BELLERS, E. B. "De-Interlacing Of Video Data," *IEEE Transactions on Consumer Electronics*, Vol. 43, No. 3, August 1997, pp. 819-825 .

[22] HAAN G. de, and. BELLERS E. B., "Deinterlacing-An Overview," *Proceedings of the IEEE*, Vol. 86, No. 9, September 1998, pp. 1839-1857.

[23] HUANG Q., GAO W., ZHAO D., and SUN H., " An Efficient and Robust adaptive Deinterlacing Technique" , *IEEE Transactions on Consumer Electronics*, Vol. 52, No. 3, August 2006, pp. 888-895.

[24] JAIN J.R. and JAIN A. K., "Displacement measurement and its application in inter frame image coding," *IEEE Transactions on Communication*, Vol. COM-29, Dec. 1981, pp.1799-1808.

[25] JEM A. J., "The Shannon sampling theorem-Its various extensions and applications: A tutorial review," *Proc. IEEE,* Vol. 65, No. 1I , Nov. 1977, pp. 1565-1596.

[26] JEON G., ANISETTI M., BELLANDI V., DAMIANI E. and JEONG J., "Fuzzy Weighted Approach to Improve Visual Quality of Edge-Based Filtering," *IEEE Transactions on Consumer Electronics*, Vol. 53, No. 4, November 2007, pp. 1661-1667

[27] JUNG Y. Y., CHOI B. T., PARK Y. J. and KO S. J., "An Effective De-Interlacing Technique Using Motion Compensated Interpolation," *IEEE Transactions on Consumer Electronics,* Vol. 46, No. 3, August 2000, pp. 460-466.

[28] JUNG Y. Y., YANG S., and YU P., "An Effective De-Interlacing Technique Using Two Types of Motion Information," *IEEE Transactions on Consumer Electronics*, Vol. 49, No. 3, August 2003, pp. 493-498.

[29] KIM W., JIN S. and JEONG J., "Novel Intra Deinterlacing Algorithm Using Content Adaptive Interpolation", *IEEE Transactions on Consumer Electronics*, Vol. 53, No. 3, August 2007, pp.1036-1043

[30] KIM Y.-T., KIM S.-H., and. PARK S.-W, "Motion decision feedback de-interlacing algorithms," in *IEEE Int. Conf. Image Processing*, 2002, pp. 397–400.

[31] KOGA T., IINUMA K., and IIJIMA A., "Motion-compensated inter frame coding for video conferencing," *Proc. NTC81*, New Orleans, LA, 1981, pp. C9.6.1-9.6.5.

[32] KOIVUNEN T., "Motion Detection of an Interlaced Video Signal," *IEEE Transactions on Consumer Electronics*, Vol. 40, No. 3, August 1994, pp. 753-760.

[33] KOVA˘CEVI´C J., SAFRANEK R. J., and YEH E. M., "Deinterlacing by Successive Approximation," *IEEE Transactions on Image Processing,* Vol. 6, No. 2 February 1997, pp. 339-344.

[34] KUO C.J., LIAO C. and LIN C.C., "Adaptive interpolation technique for scanning rate conversion," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, No. 3, June 1996, pp. 317-321.

[35] KWON O., SOHN K., and Lee C., "Deinterlacing using Directional Interpolation and Motion Compensation," *IEEE Transactions on Consumer Electronics*, Vol. 49, No. 1, February 2003, pp. 198-203.

[36] KWON S.-K., SEO K.-S., KIM J.-K., and KIM Y.-G., "A motion-adaptive de-interlacing method," *IEEE Transactions on Consumer Electronics,* Vol. 38, No. 3, pp. 145–150, Aug. 1992.

[37] LEE C., S CHANG, and C. JEN, "Motion detection and motion adaptive pro-scan conversion," in *IEEE International Symposium on Circuits and Systems,* 1991, pp. 666-669.

[38] LEE D. H., "A New Edge-based Intra-field Interpolation Method for Deinterlacing Using Locally Adaptive-threshold Binary Image," *IEEE Transactions on Consumer Electronics*, Vol. 54, No. 1, February 2008, pp. 110-115.

[39] LEE G. G., SU D. W. C., LIN H. Y. and WANG M. J., "Multiresolution-based Texture Adaptive Motion Detection for De-interlacing ", *IEEE ISCAS 2006*, pp. 4317-4320

[40] LEE H. Y., PARK J. W., BAE T. M., CHOI S. U., and HA Y. H., "Adaptive Scan Rate Up-Conversion System Based On Human Visual Characteristics", *IEEE Transactions on Consumer Electronics*, Vol. 46, No. 4, November 2000, pp. 999-1006

[41] LEE M. H., KIM J.H., RYU K. K., and SONG D.I., "A new algorithm for interlaced to progressive scan conversion based on directional correlations and its IC design," *IEEE Transactions on Consumer Electronics*, Vol. 40, No. 2, 1994, pp. 119-125.

[42] LEE S. G. and LEE D. H., "A Motion-Adaptive De-interlacing Method Using an Efficient Spatial and Temporal Interpolation, *IEEE Transactions on Consumer Electronics,* Vol. 49, No. 4, November 2003, pp.1266-1271.

[43] LI R., ZENG B., and LIOU M. L., "Reliable Motion Detection/Compensation for Interlaced Sequences and Its Applications to Deinterlacing," *IEEE Transactions on Circuits and Systems for Video Technology,* Vol. 10, No. 1, February 2000 , pp. 23-29.

[44] LI R., ZENG B., and LIOU M., "A New Three-Step Search Algorithm for Block Motion Estimation," *IEEE Trans. Circuit and Systems for Video Technology*, Vol. 4, No. 4, August 1994, pp. 438-442.

[45] LIN S.-F., CHANG Y.-L., and CHEN L.-G., "Motion adaptive interpolation with morphological operation and 3:2 pull-downed recovery for de-interlacing," in *IEEE Int. Conf. Multimedia Expo*, Lausanne, Switzerland, August 2002, CD-ROM.

[46] LIN S. F., CHANG Y. L., and CHEN L. G., "Motion Adaptive Interpolation with Horizontal Motion Detection for Deinterlacing," *IEEE Transactions on Consumer Electronics,* Vol. 49, No. 4, November 2003, pp. 1256-1265.

[47] LIN S. F., CHANG Y. L., and CHEN L. G., "Motion Adaptive De-interlacing by Horizontal Motion Detection and Enhanced ELA Processing," *ISCAS 2003*, pp. II696-II699.

[48] LIU L.K. and FEIG E., "A Block-Based Gradient Descent Search Algorithm for Block Motion Estimation in Video Coding," *IEEE Transactions on Circuit and Systems for Video Technology*, Vol. 6, No. 4, August 1996, pp. 419-422.

[49] LUO1 J., AHMAD I. and LUO X., "An Adaptive Cross Search Algorithm for Block Matching Motion Estimation," *IEEE ICCCAS*, June 2004, pp. 914-918.

[50] MAHVASH MOHAMMADI H., LANGLOIS P. and SAVARIA Y., " A Five-Field Motion Compensated Deinterlacing Method Based on Vertical Motion," *IEEE Transactions on Consumer Electronics,* Vol. 53, No. 3, August 2007, pp. 1117-1124.

[51] MAHVASH MOHAMMADI H., LANGLOIS P. and SAVARIA Y., "A Pattern-Based Directional Interpolation Deinterlacing Algorithm," *IEEE Transactions on Consumer Electronics* (Submitted).

[52] MAHVASH MOHAMMADI H., SAVARIA Y., and LANGLOIS P., "Hybrid Video Deinterlacing Algorithm Using Reverse Motion Estimation" *IEEE Transactions on Circuits and Systems for Video Technology*, (Submitted).

[53] MAHVASH MOHAMMADI H., LANGLOIS J.M. P., and SAVARIA Y., "A Threshold-Based De-Interlacing Algorithm Using Motion Compensation and

Directional Interpolation," *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Nice France, December 2006, pp. 459-462.

[54] MAHVASH MOHAMMADI H., SAVARIA Y., and LANGLOIS J.M. P., "Real Time ELA De-Interlacing with the Xtensa Reconfigurable Processor," *The 4th International IEEE-NEWCAS Conference*, June 2006, p. 25-28.

[55] NGUYEN A. and DUBOIS E., "Spatio-temporal adaptive interlaced-to-progressive conversion," in *Signal Processing of HDTV*, E. Dubois and L. Chiariglione, Eds. New York: Elsevier, 1993, pp. 749-756.

[56] PARK M. K. and KANG M. G., "New Global Motion Compensated De-interlacing Algorithm Based on Horizontal and Vertical Patterns," *ICASSP 2004*, pp. III345-III348.

[57] PARK M. K., KANG M. G., NAM K., and OH S. G., "New Edge Dependent Deinterlacing Algorithm Based on Horizontal Edge Pattern," *IEEE Transactions on Consumer Electronics*, Vol. 49, No. 4, November 2003, pp. 1508-1512

[58] PATTI A. J., SEZAN M. I., and TEKALP A. M., "Robust methods for high quality stills from interlaced video in the presence of dominant motion," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, No. 2,April 1997, pp. 328–342.

[59] PENG C., HE Z., and CAGER Y., "An Efficient Motion-adaption De-interlacing Technique on VLIW DSP Architecture", *IEEE 2005,* pp. 644-649.

[60] PIGEON S. and GUILLOTE P., Advantages and drawbacks of Interlaced and Progressive scanning formats, *CEC RACE/HAMLET Deliverable no R2110/WP2/DS/R/004/b1*, June 1995.

[61] PURI A., HANG H. M., and SCHILLING D. L., "An efficient block matching algorithm for motion compensated coding," *ICASSP'87*, Dalas, TX, 1987, pp 1063-1066.

[62] RYU C. and KIM S. P., "De-interlacing using motion compensated local spectra," in *Proc. Record 29th Asilomar Conf. Signals, Systems and Computers*, Vol. 2, 1996, pp. 1394–1397.

[63] SRINIVASAN R. and RAO K., "Predictive coding based on efficient motion estimation, " *IEEE Transactions on Communication*, Vol. COM-33, September 1985, pp. 1011-1014.

[64] SUGIYAMA K.J., and NAKAMURA H.Y., "A Method of De-Interlacing with Motion Compensated Interpolation," *IEEE Trans. Consumer Electronics*, Vol. 45, No. 3, pp. 611-616, Aug. 1999.

[65] SUN C., "De-Interlacing Of Video Images Using A Shortest Path Technique," *IEEE Transactions on Consumer Electronics,* Vol. 47, No. 2, MAY 2001, pp. 225-230.

[66] VAN DE VILLE D., ROGGE B., PHILIPS W., and LEMAHIEU I., "De-interlacing using fuzzy-based motion detection," in *Proc. 3rd Int. Conf. Knowledge-Based Intelligent Information Engineering Systems*, 1999, pp. 263–267.

[67] VAN DE VILLE D., PHILIPS W., and LEMAHIEU I., "Motion compensated de-interlacing for both real time video and still images," in *Proc. Int. Conf. Image Processing*, Vol. 2, 2000, pp. 680–683.

[68] WALLACE G. K., "The JPEG still picture compression standard," *Comm. Of the ACM*, Vol. 34, No. 4, Apr. 1991, pp 30-44.

[69] WAN F.M., ANASTASSIOU D., and NETRAVALI A. N., "Time recursive deinterlacing for IDTV and pyramid coding," *Signal Processing: Image Communication,* 2, 1990, pp. 365-374.

[70] WANG D., VINCENT A., and BLANCHFIELD P., "Hybrid De-Interlacing Algorithm Based on Motion Vector Reliability," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 15, no. 8, August 2005, pp. 1019-1025.

[71] WANG Z., BOVIK A. C., SHEIKH H. R., and SIMONCELLI E. P., " Image Quality Assessment: From Error Visibility to Structural Similarity," *IEEE Transactions on Image Processing*, Vol. 13, No. 4, April 2004, pp. 600-612.

[72] WOODS J.W. and HAN S. C., "Hierarchical Motion Compensated De-interlacing," *SPIE Visual Communication and Image Processing VI*, November 1991, Boston.

[73] XIAO R. J., GE C., LIU B., SHEN Y., and LI Y., "De-interlacing with motion compensation and edge-dependent interpolation in complement using judder pattern," *ICCE 2005*, pp. 81-82.

[74] YANG S., JUNG Y. Y., LEE Y. H., and PARK R. H., "Motion Compensation Assisted Motion Adaptive Interlaced-to-Progressive Conversion," *IEEE*

Transactions on Circuits and Systems for Video Technology, Vol. 14, No. 9, September 2004, pp. 1138-1148.

[75] YOO H. and JEONG J., "Direction-Oriented Interpolation And Its Application To De-Interlacing," *IEEE Transactions on Consumer Electronics*. Vol. 48, No. 4, November 2002, pp. 954-962.

[76] YUHONG Y., YINGQI C., WENJUN Z., "Motion Adaptive Deinterlacing Combining with Texture Detection and Its FPGA Implementation," *IEEE Int. Workshop VLSI Design & Video Tech*. May 2005, pp. 316-319.

[77] ZHAO M. and HAAN G. de, "Content Adaptive Vertical Temporal Filtering for Deinterlacing," *IEEE, ISCE 2005*, pp. 69-73.

[78] ZHU S. and MA K.K., "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation," *IEEE Transactions on Image Processing*, Vol. 9, No. 2, February 2000, pp. 287-290.

[79] http://home.arcor.de/scharfis_brain/mvbob/mvbob.rar